DATA MINING FOR SCIENTIFIC AND ENGINEERING APPLICATIONS

Data Mining for Scientific and Engineering Applications

Edited by

Robert L. Grossman
University of Illinois, Chicago

Chandrika Kamath
Lawrence Livermore National Laboratory, Livermore

Philip Kegelmeyer
Sandia National Laboratories, Livermore

Vipin Kumar
Army High Performance Computing Research Center (AHPCRC), Minneapolis

and

Raju R. Namburu
Army Research Laboratory, Aberdeen Proving Ground

SPRINGER SCIENCE+BUSINESS MEDIA, B.V.

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4020-0114-7 ISBN 978-1-4615-1733-7 (eBook)
DOI 10.1007/978-1-4615-1733-7

Printed on acid-free paper

All Rights Reserved
© 2001 Springer Science+Business Media Dordrecht
Originally published by Kluwer Academic Publishers in 2001
Softcover reprint of the hardcover 1st edition 2001
No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the copyright owner.

Table of Contents

Foreword.....ix

List of Contributors.....xi

List of Reviewers.....xvii

Preface.....xix

1 **ON MINING SCIENTIFIC DATASETS**
Chandrika Kamath.....1

2 **UNDERSTANDING HIGH DIMENSIONAL AND LARGE DATA SETS: SOME MATHEMATICAL CHALLENGES AND OPPORTUNITIES**
Jagdish Chandra.....23

3 **DATA MINING AT THE INTERFACE OF COMPUTER SCIENCE AND STATISTICS**
Padhraic Smyth.....35

4 **MINING LARGE IMAGE COLLECTIONS**
Michael C. Burl.....63

5 **MINING ASTRONOMICAL DATABASES**
Roberta M. Humphreys, Juan E. Cabanela, and Jeffrey Kriessler.....85

6 **SEARCHING FOR BENT-DOUBLE GALAXIES IN THE FIRST SURVEY**
Chandrika Kamath, Erick Cantú-Paz, Imola K. Fodor, and Nu Ai Tang.....95

7 **A DATASPACE INFRASTRUCTURE FOR ASTRONOMICAL DATA**

Robert Grossman, Emory Creel, Marco Mazzucco, and Roy Williams........115

8 DATA MINING APPLICATIONS IN BIOINFORMATICS
Naren Ramakrishnan and Ananth Y. Grama........125

9 MINING RESIDUE CONTACTS IN PROTEINS
Mohammed J. Zaki and Chris Bystroff........141

10 KDD SERVICES AT THE GODDARD EARTH SCIENCES DISTRIBUTED ARCHIVE CENTER
Christopher Lynnes and Robert Mack........165

11 DATA MINING IN INTEGRATED DATA ACCESS AND DATA ANALYSIS SYSTEMS
Ruixin Yang, Menas Kafatos, Kwang-Su Yang, and X. Sean Wang........183

12 SPATIAL DATA MINING FOR CLASSIFICATION, VISUALISATION AND INTERPRETATION WITH ARTMAP NEURAL NETWORK
Weiguo Liu, Sucharita Gopal, and Curtis Woodcock........201

13 REAL TIME FEATURE EXTRACTION FOR THE ANALYSIS OF TURBULENT FLOWS
I. Marusic, G.V. Candler, V. Interrante, P.K. Subbareddy, and A. Moss........223

14 DATA MINING FOR TURBULENT FLOWS
Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar........239

15 EVITA-EFFICIENT VISUALIZATION AND INTERROGATION OF TERA-SCALE DATA
Raghu Machiraju, James E. Fowler, David Thompson, Bharat Soni, and Will Schroeder........257

16 TOWARDS UBIQUITOUS MINING OF

DISTRIBUTED DATA
Hillol Kargupta, Krishnamoorthy Sivakumar, Weiyun Huang, Rajeev Ayyagari, Rong Chen, Byung-Hoon Park, and Erik Johnson........281

17 DECOMPOSABLE ALGORITHMS FOR DATA MINING
Raj Bhatnagar........307

18 HDDI^{TM}: HIERARCHICAL DISTRIBUTED DYNAMIC INDEXING
William M. Pottenger, Yong-Bin Kim, and Daryl D. Meling........319

19 PARALLEL ALGORITHMS FOR CLUSTERING HIGH-DIMENSIONAL LARGE-SCALE DATASETS
Harsha Nagesh, Sanjay Goil, and Alok Choudhary........335

20 EFFICIENT CLUSTERING OF VERY LARGE DOCUMENT COLLECTIONS
Inderjit S. Dhillon, James Fan, and Yuqiang Guan........357

21 A SCALABLE HIERARCHICAL ALGORITHM FOR UNSUPERVISED CLUSTERING
Daniel Boley........383

22 HIGH-PERFORMANCE SINGULAR VALUE DECOMPOSITION
David B. Skillicorn and Xiaolan Yang........401

23 MINING HIGH-DIMENSIONAL SCIENTIFIC DATA SETS USING SINGULAR VALUE DECOMPOSITION
Ekaterina Maltseva, Clara Pizzuti, and Domenico Talia........425

24 SPATIAL DEPENDENCE IN DATA MINING
James P. LeSage and R. Kelley Pace........439

25 **SPARC: SPATIAL ASSOCIATION RULE-BASED CLASSIFICATION**
Jiawei Han, Anthony K. H. Tung, and Jing He........................461

26 **WHAT'S SPATIAL ABOUT SPATIAL DATA MINING: THREE CASE STUDIES**
Shashi Shekhar, Yan Huang, Weili Wu, C.T. Lu, and S. Chawla........................487

27 **PREDICTING FAILURES IN EVENT SEQUENCES**
Mohammed J. Zaki, Neal Lesh, and Mitsunori Ogihara..............515

28 **EFFICIENT ALGORITHMS FOR MINING LONG PATTERNS IN SCIENTIFIC DATA SETS**
Ramesh C. Agarwal and Charu C. Aggarwal...........................541

29 **PROBABILISTIC ESTIMATION IN DATA MINING**
Edwin P. D. Pednault and Chidanand Apte.............................567

30 **CLASSIFICATION USING ASSOCIATION RULES: WEAKNESSES AND ENHANCEMENTS**
Bing Liu, Yiming Ma, and Ching-Kian Wong........................591

Foreword to

Data Mining for Scientific and Engineering Applications

Data Mining is automatic or semi-automatic discovery of buried knowledge in massive amounts of data. The field grew out of the need to discover useful information from ever growing data warehouses in commercial enterprises, and has roots in established disciplines such as statistics, artificial intelligence and machine learning, and pattern recognition. The scalability requirements imposed by the size of the business applications required the use of data base technology and high performance computing in data mining applications. Given the success of data mining in commercial areas, it didn't take much time for scientists and engineers to discover the usefulness of data mining techniques in scientific disciplines. For example, analysis of massive simulation data sets generated by computational simulations of physical and engineering systems is difficult and time consuming using traditional approaches. Indeed, much of the output of computational simulations is simply stored away on disks and is never analyzed at all. Availability of suitable data mining techniques can allow engineers and scientists to analyze such data and gain fundamental insights into the underlying mechanisms of the physical processes involved.

The two workshops held at the Army High Performance Computing Research Center in 1999 and 2000 were among the first steps towards exploring the promise of data mining technology to scientific and engineering domains. I commend the editors of this book for their efforts in putting together a fine volume based upon the presentations at the two workshops. The book provides a good balance between articles that cover scientific applications as well as those that cover algorithms suited for scientific and engineering applications. In addition, many of the techniques and algorithms covered in this book have applicability far beyond scientific and engineering domains.

I am confident that the book will be a valuable resource for scientists and engineers interested in exploring the use of data mining. In addition, the book will help data mining researchers to better understand the requirements of the scientific and engineering applications and motivate them to develop new techniques for such applications.

Dr. N. Radhakrishnan
Director and Corporate Information Officer
Computational and Information Sciences Directorate
US Army Research Laboratory

List of Contributors

Name	Affiliation	Email
Agarwal, Ramesh C.	IBM T. J. Watson Research Center, Yorktown Heights, NY 10598	agarwal@watson.ibm.com
Aggarwal, Charu C.	IBM T. J. Watson Research Center, Yorktown Heights, NY 10598	charu@us.ibm.com
Apte, Chid	IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598	apte@us.ibm.com
Ayyagari, Rajeev	Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, Maryland 21250	arajeev@cs.umbc.edu
Bhatnagar, Raj	Electrical and Computer Engineering and Computer Science Department, University of Cincinnati, Cincinnati, Ohio	Raj.Bhatnagar@uc.edu
Boley, Dan	Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455	boley@cs.umn.edu
Burl, Michael C.	Machine Learning Systems Group, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109	Michael.C.Burl@jpl.nasa.gov
Bystroff, Chris	Biology Department, Rensselaer Polytechnic Institute, Troy, NY 12180	bystrc@rpi.edu
Cabanela, Juan	Department of Physics, Astronomy and Engineering Science, St. Cloud State University, St. Cloud, Minnesota	juan@lua.stcloudstate.edu
Candler, Graham V.	Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN 55455	candler@aem.umn.edu
Cantu-Paz, Erick	Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551	cantupaz1@llnl.gov
Chandra, Jagdish	Department of Engineering Management and System Engineering, School of Engineering and Applied Sciences, George Washington University, Washington, D.C.	jchandra@SEAS.GWU.EDU
Chen, Rong	School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164	rchen@eecs.wsu.edu
Choudhary, Alok	Department of Electrical and Computer Engineering, Northwestern University	choudhar@ece.northwestern.edu
Creel, Emory	Laboratory for Advanced Computing, University of Illinois at Chicago	emory@lac.uic.edu

Dhillon, Inderjit S.	Department of Computer Sciences, University of Texas, Austin, TX 78712-1188	inderjit@cs.utexas.edu
Fan, James	Department of Computer Sciences, University of Texas, Austin, TX 78712-1188	jfan@cs.utexas.edu
Fodor, Imola K.	Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551	fodor1@llnl.gov
Fowler, James E.	Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS 39762	fowler@ece.msstate.edu
Goil, Sanjay	Performance Technology Group, Sun Microsystems Inc.	sanjay.goil@eng.sun.com
Gopal, Sucharita	Department of Geography and Center for Remote Sensing, Boston University, Boston, MA 02215	suchi@crsa.bu.edu
Grama, Ananth	Department of Computer Science, Purdue University, IN 47907	ayg@cs.purdue.edu
Grossman, Robert	University of Illinois at Chicago and Magnify, Inc.	grossman@uic.edu
Guan, Yuqiang	Department of Computer Sciences, University of Texas, Austin, TX 78712-1188	yguan@cs.utexas.edu
Han, Eui-Hong (Sam)	Department of Computer Science and Engineering, Army High Performance Computing Research Center, University of Minnesota	han@cs.umn.edu
Han, Jiawei	Intelligent Database Systems Research Laboratory, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6	han@cs.sfu.ca
He, Jing	Intelligent Database Systems Research Laboratory, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6	jhe@cs.sfu.ca
Huang, Weiyun	School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164	whuang@eecs.wsu.edu
Humphreys, Roberta	Astronomy Department, University of Minnesota, Minneapolis, MN 55455	roberta@aps.umn.edu
Hunag, Yan	Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455	hunagyan@cs.umn.edu
Interrante, Victoria	Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455	interran@cs.umn.edu

Johnson, Erik	Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, Maryland 21250	johnson@cs.umbc.edu
Kafatos, Menas	Center for Earth Observing and Space Research (CEOSR), School of Computing (SCS), George Mason University (GMU), Fairfax, VA 22030	mkafatos@gmu.edu
Kamath, Chandrika	Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551	kamath2@llnl.gov
Kargupta, Hillol	Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, Maryland 21250	hillol@cs.umbc.edu
Karypis, George	Department of Computer Science and Engineering, Army High Performance Computing Research Center, University of Minnesota	karypis@cs.umn.edu
Kim, Yong-Bin	National Center for Supercomputing Applications. University of Illinois at Urbana-Champaign, Champaign, IL	ybkim@uiuc.edu
Kriessler, Jeffrey	Efficient Channel Coding, Inc., Cleveland Ohio	jeffk@eccincorp.com
Kumar, Vipin	Department of Computer Science and Engineering, Army High Performance Computing Research Center, University of Minnesota	kumar@cs.umn.edu
LeSage, James P.	Department of Economics, University of Toledo, Toledo, OH 43606	jlesage@spatial-econometrics.com
Lesh, Neal	Mitsubishi Electric Research Laboratory, 201 Broadway, 8th Floor, Cambridge, MA 02139	lesh@merl.com
Liu, Bing	School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543	liub@comp.nus.edu.sg
Liu, Weiguo	Department of Geography and Center for Remote Sensing, Boston University, Boston, MA 02215	wgliu@crsa.bu.edu
Lu, C.T.	Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455	ctlu@cs.umn.edu
Lynnes, Christopher	Goddard Earth Sciences Distributed Active Archive Center, NASA/Goddard Flight Center, Greenbelt, Maryland	lynnes@daac.gsfa.nasa.gov
Ma, Yiming	School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543	maym@comp.nus.edu.sg

Machiraju, Raghu K.	395 Dreese Labs, Computer and Information Science, The Ohio State University, 2015 Neil Avenue Mall, Columbus, OH	raghu@cis.ohio-state.edu
Mack, Robert	Goddard Earth Sciences Distributed Active Archive Center, NASA/Goddard Flight Center, Greenbelt, Maryland	mack@daac.gsfc.nasa.gov
Maltseva, Ekaterina	ISI-CNR, Via P. Bucci 41C, 87036 Rende (CS)	maltseva@si.deis.unical.it
Marusic, Ivan	Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN 55455	marusic@aem.umn.edu
Mazzucco, Marco	Laboratory for Advanced Computing, University of Illinois at Chicago	marco@lac.uic.edu
Meling, Daryl D.	National Center for Supercomputing Applications. University of Illinois at Urbana-Champaign, Champaign, IL	dmeling@ncsa.uiuc.edu
Moss, Alin	Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN 55455	moss@aem.umn.edu
Nagesh, Harsha S.	Bell-Labs Research, Murray Hill, N.J.	harsha@research.bell-labs.com
Ogihara, Mitsunori	Department of Computer Science, University of Rochester, Rochester, NY 14627	ogihara@cs.rochester.edu
Pace, R. Kelly	LREC Endowed Chair of Real Estate, Department of Finance, Louisiana State University, Baton Rouge, LA 70803-6308	kelley@pace.am
Park, Byung-Hoon	Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, Maryland 21250	bpark@cs.umbc.edu
Pednault, Edwin P. D.	IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598	pednault@us.ibm.com
Pizzuti, Clara	ISI-CNR, Via P. Bucci 41C, 87036 Rende (CS)	pizzuti@deis.unical.it
Pottenger, William M.	Electrical and Engineering and Computer Science Department, Lehigh University, Bethlehem, PA	billp@eecs.lehigh.edu
Ramakrishnan, Naren	Department of Computer Science, Virginia Tech, VA 24061	naren@cs.vt.edu
Schroeder, William J.	110 Eight Street, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180	will.schroeder@kitware.com
Shekhar, Shashi	Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455	shekhar@cs.umn.edu

Sivakumar, Krishnamoorthy	School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164	siva@eecs.wsu.edu
Skillicorn, David B.	Department of Computing and Information Science, Queen's University, Kingston, Canada	skill@cs.queensu.ca
Smyth, Padhraic	Information and Computer Science, University of California, Irvine, CA 92697	smyth@ics.uci.edu
Soni, Bharat	Engineering Research Center, P.O. Box 9627, Mississippi State University, Mississippi State, MS 39762	bsoni@erc.msstate.edu
Subbareddy, Pramod K.	Department of Aerospace Engineering and Mechanics, University of Minnesota, Minneapolis, MN 55455	pramod@aem.umn.edu
Talia, Domenico	ISI-CNR, Via P. Bucci 41C, 87036 Rende (CS)	talia@deis.unical.it
Tang, Nu Ai	Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551	tang10@llnl.gov
Tung, Anthony K. H.	Intelligent Database Systems Research Laboratory, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6	khtung@cs.sfu.ca
Wang, X. Sean	Center for Earth Observing and Space Research (CEOSR), School of Computing (SCS), George Mason University (GMU), Fairfax, VA 22030	xywang@gmu.edu
Williams, Roy	California Institute of Technology	roy@cacr.caltech.edu
Wong, Ching Kian	School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543	wongck@comp.nus.edu.sg
Woodcock, Curtis	Department of Geography and Center for Remote Sensing, Boston University, Boston, MA 02215	curtis@crsa.bu.edu
Wu, Weili	Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455	wuw@cs.umn.edu
Yang, Kwang-Su	Center for Earth Observing and Space Research (CEOSR), School of Computing (SCS), George Mason University (GMU), Fairfax, VA 22030	kyang@gmu.edu
Yang, Ruixin	Center for Earth Observing and Space Research (CEOSR), School of Computing (SCS), George Mason University (GMU), Fairfax, VA 22030	ryang@gmu.edu
Yang, Xiaolan	Entrust Technologies, Ottawa, Canada	

Zaki, Mohammed J.	Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180	zaki@cs.rpi.edu

List of Reviewers

Name	Affiliation	Email
Cantu-Paz, Erick	Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551	cantupaz1@llnl.gov
Creel, Emory	Laboratory for Advanced Computing, University of Illinois at Chicago	emory@lac.uic.edu
Fodor, Imola K.	Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551	fodor1@llnl.gov
Han, Eui-Hong (Sam)	Department of Computer Science and Engineering, Army High Performance Computing Research Center, University of Minnesota	han@cs.umn.edu
Harinath, Sivakumar	Laboratory for Advanced Computing, University of Illinois at Chicago	sharinat@eecs.uic.edu
Mazzucco, Marco	Laboratory for Advanced Computing, University of Illinois at Chicago	marco@lac.uic.edu
Shekhar, Shashi	Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455	shekhar@cs.umn.edu
Sinha, Punita P.	Lawrence Livermore National Laboratory, Livermore, CA 94551	sinha2@llnl.gov
Steinbach, Michael	Department of Computer Science, University of Minnesota, Minneapolis, MN	steinbac@cs.umn.edu
Tan, Pang-Ning	AHPCRC/Department of Computer Science, University of Minnesota, Minneapolis, MN	ptan@cs.umn.edu
Turinsky, Andrei	Laboratory for Advanced Computing, University of Illinois at Chicago	turinsky@uic.edu
Wu, Weili	Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455	wuw@cs.umn.edu

Preface to

Data Mining for Scientific and Engineering Applications

Advances in information technology and high performance computing have led to massive data sets becoming available in many scientific disciplines. For example, a computational simulation running on state-of-the-art, high-performance computers can generate tera-bytes of data within a few hours, whereas human analysts may take several weeks or longer to analyze and discover useful information hidden in these data sets. Other examples of large and complex datasets arise in the fields of astronomy, medical imaging, combinatorial chemistry, bio-informatics, and remote sensing. Since the rate of production of these data far outstrips our ability to analyze them, there is an increasing interest in various scientific communities to exploit data mining techniques for the analysis of these large data sets.

Over the last few years, several international workshops have been organized to explore how data mining can complement and enhance existing methods for scientific data analysis that rely on statistics, exploratory data analysis, and domain-specific approaches. Two early workshops on Mining and Managing Massive Datasets were held in Chicago in July 1997 and San Diego in February 1998. However, the realization that a significant amount of innovative work done in the context of scientific applications was not well-represented in the mainstream data mining conferences, led us to organize a series of workshops focusing on Mining Scientific Datasets. The first workshop was held in September 1999 and the second in July 2000, both at the Army High Performance Computing Research Center (AHPCRC) in Minneapolis.

The success of these two workshops motivated us to not only continue the series of workshops, but also to capture the current work and emerging trends in a book. This collection of chapters is based on selected presentations from the first two workshops on Mining Scientific Datasets. An introductory chapter gives a broad overview of the diversity of problems in science and engineering that can be addressed through data mining, as well as the issues and challenges that differentiate scientific data mining from its commercial counterpart. The remaining chapters cover topics of general interest in data mining, applications of data mining to various problems in science and engineering, and mining algorithms that are particularly relevant for scientific and engineering data sets.

Though the focus of the book is on mining scientific data, the work described is of broader interest as many of the techniques described are equally applicable to other kinds of data, including business data, e-business data, etc. Further, the non-proprietary nature of some scientific data makes it an attractive surrogate for progress in data mining.

Putting together this book on an exciting topic was both a challenge and a pleasure. Our experiences with the series of workshops on Mining Scientific Datasets indicated that this was a growing field, replete with problems and opportunities. It also meant that some application areas, where data mining techniques have been making inroads only recently, were not included, at least in the first two workshops. As the series of workshops has evolved, new application areas have been included, leading to greater exchange of ideas among data miners, scientists, and engineers.

This collection is the work of several people. First, we would like to thank the authors of the chapters for agreeing to convert their presentations into papers and for adhering to a rather tight schedule. We would also like to acknowledge the reviewers who helped both us, and the authors, to improve their work. We thank Drs. N. Radhakrishnan at the Army Research Laboratory (ARL) and Steve Ashby at the Center for Applied Scientific Computing/Lawrence Livermore National Laboratory (CASC/LLNL) for their strong encouragement and support for holding the first two workshops on mining scientific data sets. The workshops were co-sponsored by the AHPCRC/ARL and CASC/LLNL. Mike Stewart, Peggy Stewart, and Kim Powell at the AHPCRC provided invaluable support in the organization of the workshops as well as the preparation of the manuscripts.

Robert L. Grossman
Chandrika Kamath
W. Philip Kegelmeyer
Vipin Kumar
Raju Namburu

Chapter 1

ON MINING SCIENTIFIC DATASETS

Chandrika Kamath

Abstract Data mining techniques have gained acceptance as a viable means of finding useful information in data. While the techniques can be applied to any kind of data, a brief survey of the work presented at recent conferences in data mining and knowledge discovery might lead one to believe that these techniques are being applied mainly to commercial data sets, to address problems such as customer relationship management, market basket analysis, credit card fraud, etc. Often overlooked is the fact that data mining techniques have long been applied to scientific datasets, with fields such as remote sensing, astronomy, biology, physics, and chemistry, providing a rich environment for the practice of these techniques. In this paper, I describe the various scientific and engineering areas in which data mining is playing an important role and discuss some of the issues that make scientific data mining different from its commercial counterpart. I show that the diversity of applications, the richness of the problems faced by practitioners, and the opportunity to borrow ideas from other domains, make scientific data mining an exciting and challenging field.

Keywords: Survey, data mining, scientific and engineering applications, diversity of problems, challenging field

1. Introduction

As data mining has gained acceptance in the commercial world, the important role these techniques play in the scientific and engineering domains is increasingly being overlooked. For example, a perusal of the schedule at the Knowledge Discovery and Data Mining 2000 conference is likely to give the impression that topics such as web mining, customer management, direct marketing, text mining, and electronic commerce, dominate the field today. However, this does not imply that the initial success of scientific data mining projects such as SKICAT, QuakeFinder, and JARTool [FHS96] have been ignored or worse, forgotten. To the contrary, the legacy of data analysis techniques that have been used in scientific domains for years has continued to

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 1–21.
© 2001 *Kluwer Academic Publishers.*

grow, leading to the innovative application of data mining to newer and more challenging problems.

In this chapter, I describe the diversity of scientific and engineering applications that can benefit from data mining techniques. I also discuss the issues and challenges that make the mining of scientific data different from business or commercial data. Wherever possible, I indicate the chapters in the book that further discuss an idea or application mentioned in this chapter. The chapter is organized as follows: in Section 2, I define data mining, focusing on the multidisciplinary nature of the endeavor. In Section 3, I briefly survey the numerous areas in science and engineering that lend themselves to analysis using data mining techniques. This overview of the applications does not by any means encompass all the different ways in which data mining techniques can be used - it is mainly to indicate the richness of the field and the potential benefits that can be harnessed. In Section 4, I describe the issues that make scientific data mining different from commercial data mining and discuss the challenges and opportunities that result from these differences. I conclude in Section 5 with a summary.

2. Data Mining - an overview

Several authors have observed that the term "data mining" has had a varied history [FPSS96, Smy00]. As with many emerging fields that evolve over time, borrowing ideas from other fields, data mining has had its share of terminology confusion. Some authors have viewed it as a single step in the multi-step process of KDD or Knowledge Discovery in Databases [FPSSU96], where KDD is defined as the "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data." In this viewpoint, the data mining component of KDD is seen as the means by which patterns are extracted and enumerated from the data. The other steps in KDD include tasks such as data preparation, data selection, and interpretation of the results. Other authors, for example [Sim96], view data mining as the "process of extracting valid, previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions." In other words, they see data mining, not KDD, as the overall process of extracting high-level knowledge from low-level data. Still others view data mining as nothing new, but just an extension of what they have been doing all along in their chosen field of statistics, or machine learning, or exploratory data analysis.

In what may be a wise choice, some authors [NW2], have distanced themselves from the data mining terminology debate, choosing instead to focus on the combination of techniques that characterize the field. Most practitioners and proponents of data mining do agree that regardless of how one defines the term, it is a multidisciplinary field, borrowing ideas from diverse technologies such as machine learning, statistics, databases, high performance computing, data visualization, and image processing. What is new is the confluence of the mature offshoots of these technologies at a time when we are ready to exploit them for the analysis of massive data sets. As data mining has been applied to new problem domains, this technology mix has grown as well. For example, natural language processing issues have become important in text mining, and privacy issues must be considered in electronic commerce. As we shall see in this chapter, when data mining is applied to scientific and engineering data, it adds new technologies to this multidisciplinary field.

3. Mining Scientific and Engineering data

There is a wide range of problems and application domains in science and engineering that can benefit from data mining. In several of these fields, techniques similar to data mining have been used for many years, albeit under a different name. For example, in the area of remote sensing, rivers and boundaries of cities have been identified using image understanding methods. Much of the use of data mining techniques in the past has been for data obtained from observations or experiments, as one-dimensional signals or two-dimensional images. However, these techniques are increasingly attracting the attention of scientists involved in simulating complex phenomena on massively parallel computers. They realize that, among other benefits, the semi-automated approach of data mining can complement visualization in the analysis of massive datasets produced by the simulations.

I next describe the multi-faceted role that data mining can play in various areas in science and engineering. The intent is to give an idea of the diversity of problems that can be addressed and the richness of the field. It is by no means an exhaustive survey of the work that has been done in each of the problems, nor is it intended to cover all the possible applications of data mining in a problem domain or all problem domains with large datasets.

There are different data types that arise in scientific applications. These can be broadly categorized as follows:

- One-dimensional data: These are typically signals collected by single or multiple sensors, often available as a time series.

- Two-dimensional data: This is data available as images (a regular grid of pixels) or mesh data arising from complex simulations in two dimensions. The mesh data can be broadly categorized into structured and unstructured data. In structured data types, the data can be either regular in the form of a Cartesian mesh, Figure 1, or curvilinear as shown in the left panel in Figure 2. In the case of a regularly-spaced Cartesian mesh, techniques from image processing can be applied to pre-process the mesh data. In the unstructured mesh data category, for example the right panel in Figure

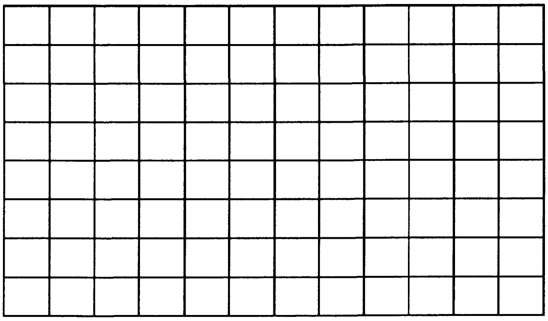


Figure 1. An example of an image or a regularly-space Cartesian mesh

2, the data can be composed of either homogeneous elements (in this case triangles) or heterogeneous elements as in Figure 3. In addition to structured and unstructured meshes there are also meshes that are the hybrid of the two - these include hierarchical meshes which are globally unstructured but locally structured, Figure 4, as well as meshes that are structured in one part of the problem domain and unstructured in another.

In many problems, the meshes are not static - as the physical features in the simulation evolve, the mesh too must adaptively change to accurately model these features. There are also mesh-less methods used in simulations that result in output data that is irregularly spaced. Often, a series of two-dimensional data sets are produced as the simulation evolves in time and data is output at each time step. This is also true for image data, where one might consider a data cube obtained by stacking images taken at different times. Thus, in both image and mesh data, there is a spatial and temporal aspect that must be considered in the analysis.

- Three-dimensional data: This typically arises from simulations in three dimensions. Again, the mesh data can be structured or unstructured. As in the two-dimensional case, the simulation may evolve in time, creating as output, a series of three-dimensional data sets. Both spatial and temporal aspects must be considered in this case as well.

I next list the areas which provide an opportunity for the use of data mining and briefly describe the problems that are being addressed in each field. Some of these areas overlap and a problem may appear more than once.

- **Astronomy:** The fields of astronomy and astrophysics have long used data analysis techniques. Since observations form an important component of this science, technologies such as statistics, that aid in the

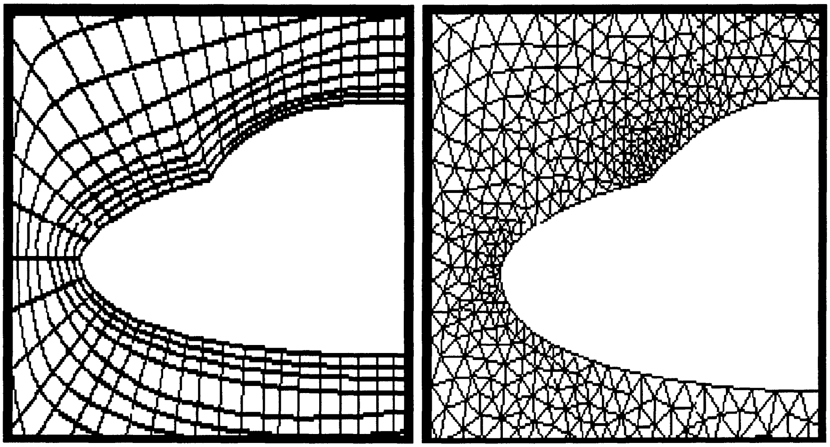


Figure 2. An example of two-dimensional structured mesh (left) and unstructured mesh (right) around the front of an aircraft (http://www.nas.nasa.gov/Pubs/Docs/FAST/chp_16.surferu.html)

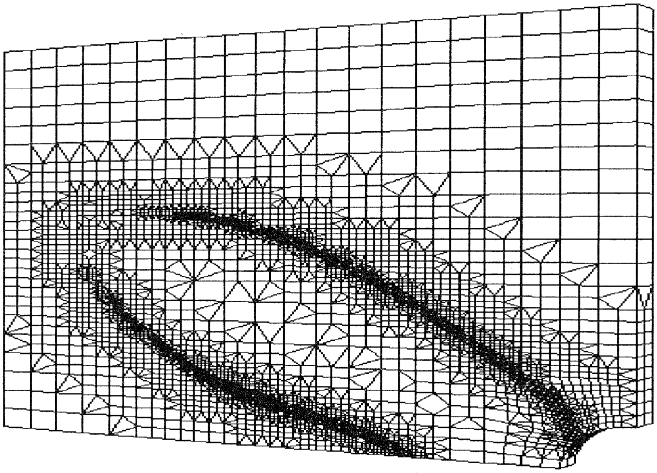
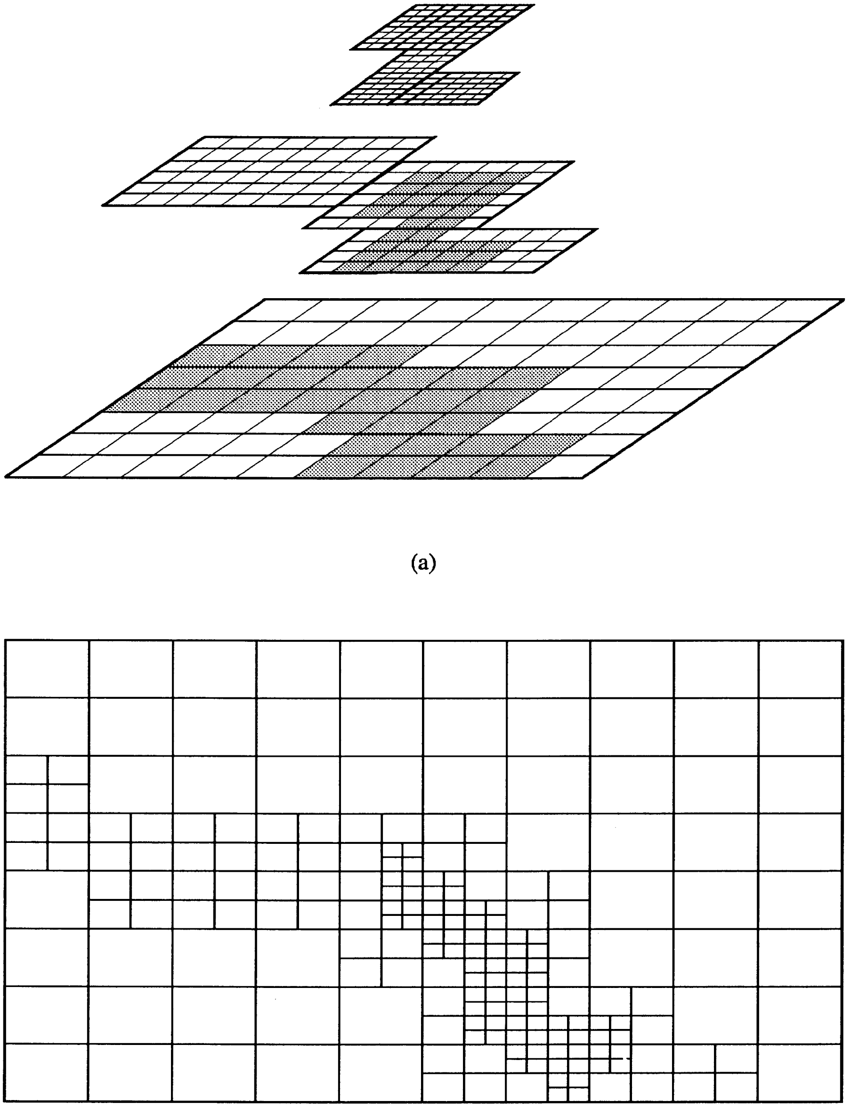


Figure 3. An example of a three-dimensional unstructured mesh with heterogeneous elements (http://cox.iwr.uni-heidelberg.de/ ug/Images/benchmark_grid.gif)



(a)

(b)

Figure 4. An example of a composite mesh made up of regular meshes illustrating (a) the hierarchy of regular meshes and (b) the composed "unstructured mesh". (http://www.llnl.gov/casc/samrai)

careful interpretation of the observations, are an integral part of astronomy. With the advent of new telescopes, detectors, and CCD cameras, the data being collected from astronomical surveys is now being measured in terabytes. Further, with higher performance computers and easy availability of storage, it is becoming possible to store and analyze these datasets. Astronomy data can be either time series data, image data, or a series of image data. The data items typically number in the hundreds of thousands to several million. The number of features or attributes is often high as well, often numbering in the hundreds. Further, astronomy data is usually not stored in databases.

The SKICAT and JARTool projects from the Jet Propulsion Laboratory are the more well known projects in mining astronomy data, where the problems addressed were the classification of faint objects and the detection of volcanos on Venus, respectively [FSBP96, BAS$^+$98]. These projects approached the problem mainly from a data mining perspective. However, earlier work done by astronomers [JT81, OSP$^+$92, SLLSSL92] also used data mining techniques such as image processing, non-parametric statistical classifiers, and neural networks for problems such as morphological classification, discrimination of stars and galaxies, and the analysis of faint objects. More recent work by both astronomers and data miners [MC96, BFR$^+$99, NH99] has focused on the solution of astronomical problems as well as the more practical issues of data pre-processing and supporting a software infrastructure for distributed client-server architectures. These issues are discussed further in three chapters in this book - "Mining Large Image Collections" by Michael Burl, "Mining Astronomical Databases" by Roberta Humphreys, Juan Cabanela, and Jeff Kriessler and "Searching for Bent-Double Galaxies in the FIRST Survey", by Chandrika Kamath, Erick Cantú-Paz, Imola K. Fodor, and Nu Ai Tang. The work presented in these three chapters are a clear indication that this collaboration between the data mining and the astronomy communities has been mutually beneficial.

While much of the focus in mining astronomy data is on data obtained through observations, it is also possible to mine data from computer simulations of astronomical phenomena. With compute power becoming easily available, it is now possible to mathematically model the behavior of complex phenomena such as the evolution of a star. The datasets that result from these computer simulations are typically analyzed by means of visualization techniques, though the massive size of many simulations are prompting astrophysicists to actively consider data mining techniques.

The challenges and opportunities presented by astronomy data are likely to increase as the Virtual Observatory [NVO] goes on-line. Some early work in creating virtual observatories using an Internet infrastructure is presented in the chapter on "A DataSpace Infrastructure for Astronomical Data" by Robert Grossman, Emory Creel, Marco Mazzucco, and Roy Williams.

Astronomy data presents several unique challenges. For example, there is frequently noise in the data due to the sensors used for collecting the data, atmospheric disturbances, etc. The data may also be corrupted with missing values or invalid measurements. In the case of images, identifying an object within an image may be non-trivial as natural objects are frequently complex and image processing techniques based on the identification of edges or lines are inapplicable. Further, the raw data, which is in a high-dimensional space, must be transformed into a lower dimensional feature space, resulting in expensive pre-processing. The volumes of data are also very large, further exacerbating the problem. All these characteristics, in addition to the lack of ground truth, make astronomy a challenging field for the practice of data mining.

- **Biology, Chemistry, and Medicine:** Bioinformatics, chemical informatics, and medicine are all areas where data mining techniques have been used for a while and are increasingly gaining acceptance.

  In bioinformatics, which is a bridge between biology and information technology, the focus is on the computational analysis of gene sequences. Here, the data can be either gene sequences, expressions (information on how the different parts of a sequence are activated), or protein data (representing the biochemical and biophysical structure of the molecules). In light of the recent work in the sequencing of the human genome, research in bioinformatics will evolve from analyzing the effects of individual genes to a more integrated view that examines whole ensembles of genes as they interact to form a living human being [Con01, VAM01]. This active area is discussed further in the chapter on "Data Mining Applications in Bioinformatics" by Naren Ramakrishnan and Ananth Y. Grama.

  In contrast with most scientific data sets that are stored as flat files, there are several databases for both sequence and protein data. As a result, infrastructure issues related to the integration of databases and flexible access to the data are a challenging problem in bioinformatics [BZ99]. The data mining tasks for biosequences include finding the genes in the DNA sequences and understanding the higher order structure of a protein in order to understand the function of the protein [FHS96]. The former task typically uses techniques such as neural networks, while the latter tasks of protein structure prediction and modeling are usually performed using Hidden Markov Models [ZJB00]. The topic of predicting contact potentials among protein residues or amino acids is explored further in the chapter on "Mining Residue Contacts in Proteins" by Mohammed Zaki and Chris Bystroff.

In addition to gene-related data, biological data is often available in the form of images as well. For example, protein crystallography is among the most successful methods for determining the three-dimensional conformation of a protein. However, this is a labor-intensive method and in order to bring automation to the process, scientists are applying techniques from scene analysis in computer vision. An example of this is given in [LGB$^+$97], where the technique of critical points is used to identify relevant features in the images, followed by a Bayesian approach to identify secondary structures. It is interesting to note that the techniques used in this case, as well as the challenges, are the same as the ones in remote sensing and machine vision. Another area where image mining is used in medicine is the analysis of images from mammograms, MRI scans, ultrasound, DNA microarrays, and X-rays for tasks such as identifying tumors, retrieving images with similar characteristics, detecting changes, and genomics. Also, one cannot ignore the use of data mining in the analysis of medical records, an area where statistics has had a long history.

In the chemical sciences, the information overload problem is becoming staggering as well, with the Chemical Abstracts Service adding 700,000 new compounds to its database each year [CIW]. Chemistry data is usually obtained either by experimentation or by computer simulation. The need for effective and efficient data analysis techniques is also being driven by the relatively new field of combinatorial chemistry, which essentially involves reacting a set of starting chemicals in all possible combinations, thus producing large data sets. Data mining is being used to analyze chemical datasets for molecular patterns and to identify systematic relationships between various chemical compounds. One of the key problems in chemical data mining is an adequate descriptor to represent the chemical structures; several potential solutions have been suggested [ZG99]. The application of neural networks and genetic algorithms to chemical data analysis is another active area of research [HG].

The diversity and the richness of the data mining tasks in biology, chemistry, and medicine is matched only by the enormous potential payoffs of success, with better drug and vaccine design, drug discovery, a more objective identification of tumors, and a better understanding of the Human Genome.

- **Earth Sciences, climate modeling, and remote sensing:** Another field replete with data mining opportunities is the area of earth sciences. This field covers a broad range of topics including climate modeling and analysis, atmospheric sciences, geographical information systems, and remote sensing. As in the case of astronomy, this is another area where the vast volumes of data have resulted in the use of semi-automated techniques for data analysis. Consequently, substantial effort has already been put into setting up a good infrastructure for data mining [NW1, NW2, RCGK00]. The practical issues that must be addressed in processing, storing, and distributing earth science data are described in the chapters on "KDD Services at the Goddard Earth Sciences Distributed Active Archive Center" by Christopher Lynnes and Robert Mack, and "Data Mining in Integrated Data Access and Data Analysis Systems" by Ruixin Yang, Menas Kafatos, Kwang-Su Yang, and X. Sean Wang. In addition, the chapter on "Spatial Data Mining for Classification, Visualization, and Interpretation with ARTMAP Neural Network" by Weiguo Liu, Sucharita Gopal, and Curtis Woodcock, discusses how the ARTMAP neural network can be integrated with an image processing software to perform automated classification of remote sensing data.

Earth science data can be particularly challenging from a practical viewpoint as it comes in many different formats, scales and resolutions. Extensive work is needed to pre-process the data, including image processing, feature extraction and feature selection. It goes without saying that the volumes of earth sciences data are typically enormous, with the NASA Earth Observing System expected to generate over 11,000 Terabytes of data on completion [FHM96, EOS]. Much of this data is stored in flat files, not databases.

The types of problems addressed by data mining in earth sciences data include understanding ozone depletion and greenhouse effects, desert and vegetation patterns, and land and ocean productivity. These in turn help us to understand the inter-relationships between the earth's geosphere, biosphere, atmosphere, and cryosphere. Other areas where data mining is being applied include land cover classification for monitoring change, planning and assessing land-use impact; detection of earthquakes from space; forest management; cloud detection; and early storm warning [SCC$^+$95, SD96, RCGK00].

Earth science data is available not only as images from satellites looking down on the earth, but also in the form of observations obtained on, above, or under the earth's surface, as well as mesh data obtained from the simulation of the earth's climate [SNM$^+$95]. It must be noted that earth science data from observations tends to be incomplete, both spatially and temporally.

- **Computer Vision and Robotics:** There is a substantial overlap between the fields of computer vision and robotics on one hand, and data mining on the other [CVW]. There are several ways in which the two fields can benefit each other. For example, computer vision applications can benefit from the accurate machine learning algorithms developed in data mining, while the extensive work done in image analysis and fuzzy logic for computer vision and robotics can be used in data mining especially for applications involving images.

The applications of data mining methodologies in computer vision and robotics are very diverse. They include automated inspection in industries for tasks such as detecting errors in semiconductor masks and identifying faulty widgets in assembly line production; face recognition and tracking of eyes, gestures, and lip movements for problems such as lip-reading, automated television studios, video conferencing, and surveillance; medical imaging during surgery as well as for diagnostic purposes; and vision for robot motion control. One of the key characteristics of problems in computer vision and robotics is that they must be done in real time. In addition, the data collection and analysis can be tailored to the task being performed as the objects of interest (e.g. widgets on an assembly line) are likely to be similar to each other.

- **Engineering:** With sensors and computers becoming ubiquitous and powerful, and engineering problems becoming more complex, there is a greater focus on gaining a better understanding of these problems through experiments and simulations. As a result, large amounts of data are being generated, providing an ideal opportunity for the use of data mining techniques in areas such as structural mechanics, computational fluid dynamics, material science, and the semi-conductor industry.

  Data from sensors is being used to address a variety of problems including detection of land mines, identification of damage in aerodynamic systems or structures such as helicopters and bridges, and nondestructive evaluation in manufacturing quality control, to name just a few [FS96, NAV97]. It is clear that sensor-based intelligent systems are here to stay, along with the accompanying need for data analysis. Note that there is considerable overlap between computer vision applications and engineering applications that work with data from sensors.

  In computer simulation, which is increasingly seen as the third mode of science, complementing theory and experiment, the techniques from data mining are yet to gain a widespread acceptance. However, some on-going projects, especially those studying the identification of coherent structures in turbulence, have started exploring the use of data mining techniques in their work [FGRAG96, CIT, TFW]. This subject has been discussed further in the chapters on "Data Mining Tools for the Analysis of Turbulent Flows: Needs, Issues, and Potential" by Ivan Marusic, G.

V. Candler, V. Interrante, P. K. Subbareddy, and A. Moss, and " Data Mining for Turbulent Flows" by Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. Other projects [TB97, PK99, SG00, Par00] are using components of data mining for various simulation-related tasks such as genetic algorithms and neural networks for mesh partitioning, principal component analysis and neural networks to model miscible fluid displacement, and evolutionary algorithms for design and manufacture.

Other applications of data mining in engineering include the analysis of simulation output as a complementary technique to visualization. The efficient visualization of massive data sets is described in the chapter on "EVITA - Efficient Visualization and Interrogation of Tera-Scale Data" by Raghu Machiraju, James Fowler, David Thompson, Bharat Soni, and Will Schroeder. Integrating data mining within such a framework for visualization will be a particularly challenging task.

- **Security and Surveillance:** This is another active area for data mining methodologies. It includes applications such as fingerprint and retinal identification, human face recognition, and character recognition in order to identify people and their signatures for access, law enforcement or surveillance purposes. In addition, tasks such as automated target recognition also use techniques from data mining [NAV97]. As in computer vision tasks, real time results are important in security and surveillance [Gro92, CC93].

The above list barely scratches the surface in the variety of scientific and engineering problems that can benefit from data mining. I do not even claim to have listed all the various technological areas that produce enormous amounts of data. For example, high energy physics data from particle physics experiments are likely to exceed a petabyte per year, and data from the instrumentation of computer programs run on massively parallel machines are too voluminous to be analyzed manually. However, what is becoming clear is that the data analysis problems in science and engineering are getting more complex and more pervasive, giving rise to a wonderful opportunity for the application of data mining methodologies. As I shall describe in the next section, we need to pay attention to the unique requirements of these datasets and tailor the data mining process to address these issues.

4. Challenges specific to scientific data

As I have described in earlier sections, data sets that arise in various physical sciences and engineering applications are usually the result of computer simulations, observations, or experiments. These data are typically in the form of signals (including images) or mesh data. I next discuss the ways in which mining scientific data differs from its commercial counterpart and the challenges that result from these differences.

- **Feature extraction from low-level data:** Scientific data used in analysis is often in the form of images or mesh data in two and three dimensions, involving several variables. This data is typically available in a raw form, with values at each pixel in an image, or each grid point in a mesh. As the patterns of interest are at a higher level, additional features must be extracted from the raw data prior to pattern recognition. This feature extraction task can be rather complicated as the features to be extracted are dependent on the problem being solved. Often, the extraction of features becomes an iterative task, with an initial set being refined if the results of the data mining are not satisfactory. Moreover, if the same data is used for a different problem, as is often the case in science, a different set of features might have to be extracted. When the data is available in the form of mesh data, the complex spatial structure, especially in three dimensions, poses a unique problem for the extraction of features. As the mesh might evolve over time, changing the spatial structure, this feature extraction may have to be repeated each time step. Adding to the challenge is the fact that a good knowledge of the scientific domain and the problem are required to identify and extract features in a robust manner. These issues are discussed in several chapters in this book.

- **Noisy data:** Scientific data, especially data from observations and experiments, is noisy. This noise may vary within an image, from image to image, and from sensor to sensor. Removing the noise from data, without affecting the signal, is a challenging problem in scientific data sets. Furthermore, in the case of missing or invalid data, the replacement data values must be scientifically justified, which is a non-trivial task.

- **Images with "real" data:** Often, the data available as images in science are not available as gray scale values, but real values, that is, the pixels represent some real physical quantity such as intensity of radio waves. Care must be taken to preserve this physical nature of the data in any preprocessing.

- **Size and type of the data:** Science data sets range from moderate to massive, with the largest being measured in Terabytes. As more complex simulations are performed, and observations over longer periods at higher resolutions are conducted, the data is expected to grow to the Petabyte range. Since science data has both a spatial and temporal aspect, storing and accessing these large datasets is rapidly becoming an issue. Further, from a practical viewpoint, any data mining infrastructure must support the variety of data formats that are used in each scientific domain.

- **Need for data fusion:** Frequently, scientific data is collected from various sources, using different sensors. In order to use all available data to enhance the analysis, we need data fusion techniques. This is a difficult problem if the data is collected at different resolutions, using different wavelengths, under different conditions, with different sensors. This issue arises not just in observational data, but also when simulation data is compared with experimental data for validation.

- **Lack of labeled data:** Unlike commercial data where labeled examples can be generated historically, in science data, these examples are usually generated manually. This tedious process is made more complicated as not all scientists may agree on a label for an object, or may want the data mining algorithm to identify "interesting" objects, not just objects that are similar to the training set. Further, in some domains, such as astronomy, there is no ground truth that can be used to evaluate the accuracy of data mining techniques.

- **Data in flat files, not data bases:** Unlike commercial data, scientific data is rarely available in a cleaned state in data warehouses. There are of course exceptions; for example, bioinformatics data is usually in databases and some domains store meta-data in databases and the data itself in flat files.

- **Mining data as it is being generated:** In the case of simulation data, scientists are interested in the behavior of the scientific phenomena as it changes with time. Sometimes, the time taken to output the result of the simulation at each time step may even exceed the simulation time itself. Since the simulations are run on large parallel computers, with hundreds to thousands of processors, it may be possible to perform some of the pre-processing while the data is being generated, resulting in a smaller output. Mining data on-the-fly is also desired when scientists want to monitor a long-running simulation for correctness. While this idea appears simple, a practical implementation is non-trivial.

- **Privacy issues:** Typically data miners working on scientific data do not have to address privacy or security issues that are common in many commercial datasets. There are exceptions, of course; for example, patient medical data, or data used for surveillance purposes. Some scientific data data in areas such as astronomy or earth sciences are freely accessible on the web; however, a close collaboration with a domain scientist is essential to identify relevant problems, and to understand and mine the data.

- **Growing datasets:** Many datasets in science are analyzed as they are acquired. This is typical in astronomy, where a part of the survey is conducted each year, and in simulations, where a simulation might run for weeks. In such cases, the data mining infrastructure must accommodate these growing datasets easily.

- **Real-time turnaround:** As observed, in some scientific problem domains such as computer vision and surveillance, real time turnaround is needed. In this respect, problems in science data are not unlike problems in commercial data, such as detection of credit card fraud.

While mining science and engineering data leads to some unique challenges that are different from those faced in mining commercial data, there are some challenges that are common to both types of data. These include:

- **Mining distributed data:** We are increasingly mining data that is distributed, whether it is text data that is distributed across various web servers or astronomy data that is distributed as part of a virtual observatory. Several authors in this book have addressed this issue of mining distributed data. These include Raj Bhatnager in his chapter on "Decomposable Algoruthms for Data Mining", Hillol Kargupta et. al in their chapter on "Towards Ubiquitous Mining of Distributed Data", and Bill Pottenger, Yong-Bin Kim, and Daryl D. Meling in their chapter on "HDDI: Hierarchical Distributed Dynamic Indexing".

- **Need for scalable algorithms:** With the increasing size of the data, there is a growing appreciation for algorithms that are scalable. Scalability typically refers to the ability to use additional resources, such as CPU and memory, in an efficient manner to solve increasingly larger problems. In addition, scalability can also describe how the computational requirements of an algorithm grow with problem size, an issue that can be discussed independent of the computing platform. Both these aspects have been addressed in this book. For example, Harsha Nagesh, Sanjay Goil, and Alok Choudhary exploit parallelism as an approach to scalability in their chapter on "Parallel Algorithms for Clustering High-Dimensional Large-scale Data sets". On the other hand, the chapters on "A Scalable Hierarchical Algorithm for Unsupervised Clustering" by Daniel Boley and "Efficient Clustering of Very Large Document Collections" by Inderjit Dhillon, James Fan, and Yuqiang Guan, discuss techniques that are efficient even on a single processor.

- **High dimensional data sets:** Data sets obtained from a variety of contexts such as the dynamics of the internet, remote sensing, gene sequencing etc. are typically high dimensional, with attributes numbering from a few hundred to the thousands. For the subsequent processing to be tractable, it is important that the dimension of these data sets be reduced.

The chapter by Jagdish Chandra on "Understanding High Dimensional and Large Data Sets: Some Mathematical Challenges and Opportunities" summarizes the issues and challenges that result from the high dimensionality of these data sets. David Skillicorn and Xiaolan Yang, in their chapter on "High-Performance Singular Value Decomposition" present several parallel algorithms for reducing the dimensionality of data sets typically encountered in various data mining applications. The issues of high dimensional data is also addressed by Ekaterina Maltseva, Clara Pizzuti, and Domenico Talia in their chapter on "Mining High-Dimensional Scientific Data Sets Using Singular Value Decomposition".

- **Spatial dependence of the data:** Problems in areas such as remote sensing, geographic information systems, GPS systems, etc. typically have a spatial dependence to them. This spatial aspect results in more complex models and the resulting computational requirements have traditionally prevented the application of spatial models in data mining. In their chapter on "Spatial Dependence in Data Mining", James LeSage and R. Kelly Pace discuss recent advances that allow the incorporation of spatial information in data mining models. A different approach to the classification of spatial objects is described in the chapter on "SPARC: Spatial Association Rule-based Classification" by Jiawei Han, Anthony Tung, and Jing He, who use association rules to better understand relationships in spatial databases. Shashi Shekhar et. al, in the chapter on "What's Spatial about Spatial Data Mining: Three Case Studies", observe that spatial data mining is a field in its infancy, where the typical problems and the corresponding solution frameworks still need to be defined and provided. When this spatial aspect is combined with a temporal aspect, as is often the case in many scientific applications, the search for patterns can become very computationally expensive and difficult to model accurately.
- **Search for rare or long patterns:** In some application domains, such as network intrusion detection, or finding rare patterns in DNA sequences, the classification problem can have a highly skewed distribution, where the item of interest occurs rarely. Mohammed Zaki, Neal Lesh, and Mitsunori Ogihara, in their chapter on "Predicting Failures in Event Sequences" describe an algorithm that can effectively prune out unpredictive and redundant patterns in such cases. A different problem is addressed by Ramesh Agarwal and Charu Aggarwal in their paper on "Efficient Algorithms for Mining Long Patterns in Scientific Data Sets", where they focus on efficient techniques for long patterns that occur in data sets with relatively few records or instances.
- **Building reliable and accurate models:** A fact often overlooked in data mining as we work with increasingly complex, high-dimensional, mas-

sive data sets, is that the models we build from the data must reflect the empirical characteristics of the observed data. In other words, we must ensure that the statistical assumptions of the modeling technique used in data mining match the statistical properties of the data itself. Otherwise, the models can be unreliable. This issue is explored further in the chapter by Edwin Pednault and Chidanand Apte on "Probabilistic Estimation in Data Mining", who borrow ideas from the field of robust estimation to build more reliable models. A similar sentiment is also expressed by Padhraic Smyth in his chapter on "Data Mining at the Interface of Computer Science and Statistics", as he discusses the negative connotation that some statisticians associate with the term "data mining". In addition to reliable models, we also need models which are accurate. This issue is explored in several chapters, including the one by Bing Liu, Yiming Ma, and Ching Kian Wong on "Classification Using Association Rules: Weaknesses and Enhancements".

In light of the above factors, it is clear that mining science and engineering data can be challenging. As data mining is being applied to new types of data such as mesh data from simulations, new issues are being encountered. Several practitioners, including many authors in this book, have commented on the lessons they have learned from mining science data, [FS95, FHS96, FHM96, FS96, FSBP96, LGB$^+$97, BAS$^+$98, BZ99, RCGK00]. While much remains to be done, data mining for science and engineering data is a technically challenging field where opportunities abound.

5. Summary

In this chapter, I described the diversity of the tasks that can be addressed through the use of data mining in science an engineering applications. As we have observed, this is a very rich field, replete with challenges and opportunities. As data mining techniques become an integral part of scientific data analysis, I hope that scientists and engineers will no longer echo astronomers who have voiced concerns that the sheer size of their datasets has resulted in a loss of serendipitous discoveries that were so vital to progress in the area in the past.

Acknowledgments

I gratefully acknowledge the members of the Sapphire project team for thought-provoking discussions and Sisira Weeratunga for information on the mesh data.

UCRL-JC-141709: This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[BAS^{+}98] M. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler, and J. Aubele. Learning to recognize volcanos on Venus. *Machine Learning*, 30:165–195, 1998.

[BFR^{+}99] M. Burl, C. Fowlkes, J. Roden, A. Stechert, and S. Mukhtar. Diamond Eye: A distributed architecture for image data mining. In *Data Mining and Knowledge Discovery, Proceedings of the SPIE*, April 1999.

[BZ99] V. Brusic and J. Zeleznikow. Knowledge discovery and data mining in biological databases. *The Knowledge Engineering Review*, 14(3):257–277, 1999.

[CC93] G. Candela and R. Chellappa. Comparative performance of classification methods for fingerprints. Technical Report Technical Report NISTIR 5163, National Institute of Standards and Technology, April 1993.

[CIT] Stanford University, Center for Integrated Turbulence Simulations. http://ctr-sgi1.stanford.edu/CITS/.

[CIW] Indiana University Chemical Informatics Web page. http://www.indiana.edu/~cheminfo/informatics/cinform_what-is.html.

[Con01] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[CVW] The Computer Vision Homepage at Carnegie Mellon University. http://www.cs.cmu.edu/afs/cs/project/cil/www/vision.html.

[EOS] NASA's Earth Observing System Web page. http://eospso.gsfc.nasa.gov/eospso_homepage.html.

[FGRAG96] J. Ferre-Gine, R. Rallo, A. Arenas, and F. Giralt. Identification of coherent structures in turbulent shear flows with a fuzzy ARTmap neural network. *International Journal of Neural Systems*, 7(5):559–568, November 1996.

[FHM96] G. Fargion, R. Harberts, and J. Masek. Emerging technology becomes an opportunity for EOS earth observing system. In *SPIE Proceedings*, volume 2820, pages 299–304, 1996.

[FHS96] U. Fayyad, D. Haussler, and P. Stolorz. KDD for science data analysis: issues and examples. In *Knowledge Discovery and Data Mining*, pages 50–56. AAAI Press, 1996.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Commu-*

nications of the ACM, Special Issue on Data Mining, 39:27–34, 1996.

[FPSSU96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[FS95] U. Fayyad and P. Smyth. From massive data sets to science catalogs: applications and challenges. In *Proceedings of the Workshop on Massive Data Sets*, 1995.

[FS96] W. Faller and S. Schreck. Neural networks: applications and opportunities in aeronautics. *Prog. Aerospace Scr.*, 32:433–456, 1996.

[FSBP96] U. Fayyad, P. Smyth, M. Burl, and P. Perona. Learning to catalog science images. In *Early Visual Learning*, pages 237–268. Oxford University Press, 1996.

[Gro92] P. Grother. Karhunen-Loeve feature extraction for neural handwritten character recognition. In *Applications of Artificial Neural Networks, SPIE Proceedings*, volume 1709, pages 155–166, 1992.

[HG] M. Hemmer and J. Gasteiger. Data mining in chemistry. Terena Networking Conference, 2000. See web page at http://www.terena.nl/tnc2000/proceedings/10B/10b5.html.

[JT81] J. Jarvis and J. Tyson. FOCAS: Faint object classification and analysis system. *The Astronomical Journal*, 86(3):476–495, 1981.

[LGB$^+$97] L. Leherte, J. Glasgow, K. Baxter, E. Steeg, and S. Fortier. Analysis of three-dimensional protein images. *Journal of Artificial Intelligence Research*, 7:125–159, 1997.

[MC96] A. Miller and M. Coe. Star/galaxy classification using Kohonen self-organizing maps. *Mon. Not. R. Astron. Soc.*, 279:293–300, 1996.

[NAV97] Technology for the United States Navy and Marine Corps, 2000-2035: Becoming a 21st-Century Force, 1997. National Academy of Sciences, http://www.nap.edu/html/tech_21st/iwindex.htm.

[NH99] M. Ng and Z. Huang. Data-mining massive time series astronomical data: challenges, problems and solutions. *Information and Software Technology*, 41:545–556, 1999.

[NVO] National virtual observatory. http://www.srl.caltech.edu/nvo.

[NW1] NASA workshop on data fusion and data mining. NASA Ames Research Center, Moffett Field, CA. July 9-10, 1999. See the workshop agenda at http://ic.arc.nasa.gov/ic/data99-workshop/.

[NW2] NASA workshop on issues in the application of data mining to scientific data. University of Alabama in Huntsville, October 19 - 21, 1999. See the workshop report at http://www.cs.uah.edu/ thinke/NASA_Mining/.

[OSP+92] S. Odewahn, E. Stockwell, R. Pennington, R. Humphreys, and W. Zumach. Automated star/galaxy discrimination with neural networks. *The Astronomical Journal*, 103(1):318–331, 1992.

[Par00] I. Parmee, editor. *Evolutionary Design and Manufacture*. Springer, 2000.

[PK99] D. Pham and D. Karaboga. *Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer, 1999.

[RCGK00] R. Ramachandran, H. Conover, S. Graves, and K. Keiser. Challenges and solutions to mining earth science data. In *Data Mining and Knowledge Discovery, SPIE Proceedings*, volume 4057, pages 259–264, 2000. See also the web page at http://datamining.itsc.uah.edu/.

[SCC+95] N. Short, R. Cromp, W. Campbell, J. Tilton, J. LeMoigne, G. Fekete, N. Netanyahu, W. Ligon, and K. Wichmann. Mission to planet earth: AI views the world. *IEEE Expert/Intelligent Systems and Their Applications, Special issue on environmental applications of AI*, 10(6), December 1995.

[SD96] P. Stolorz and C. Dean. Quakefinder: A scalable data mining system for detecting earthquakes from space. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 208–213. AAAI Press, 1996.

[SG00] N. Smaoui and R. Gharbi. Using Karhunen-Loeve decomposition and artificial neural network to model miscible fluid displacement in porous media. *Applied Mathematical Modeling*, 24:657–675, 2000.

[Sim96] E. Simoudis. Reality check for data mining. *IEEE Expert*, 11(5):26–33, October 1996.

[SLLSSL92] M. Storrie-Lombardi, O. Lahav, L. Sodre, and L. Storrie-Lombardi. Morphological classification of galaxies by artificial neural networks. *Mon. Not. R. Astron. Soc.*, 259:8–12, 1992.

[Smy00] P. Smyth. Data mining: data analysis on a grand scale? *Statistical Methods in Medical Research*, pages 309–327, August 2000.

[SNM+95] P. Stolorz, H. Nakamura, E. Mesrobian, R. Muntz, E. Shek, J. Santos, J. Yi, K. Ng, S. Chien, C. Mechoso, and J. Farrara. Fast spatio-temporal data mining of large geophysical datasets. In

Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pages 300–305. AAAI Press, 1995.

[TB97] B. Topping and A. Bahreininejad. *Neural Computing for Structural Mechanics*. Saxe-Coburg Publications, 1997.

[TFW] Dynamic feature extraction and data mining for the analysis of turbulent flows: Project web page. http://www.aem.umn.edu/research/marusic_nsf_dyn.html.

[VAM01] N. C. Venter et. al. The Sequence of the Human Genome. *Science*, 291:1304 - 1351, 2001.

[ZG99] J. Zupan and J. Gasteiger. *Neural networks in chemistry and drug design*. Wiley-VCH, 1999.

[ZJB00] M. Zaki, S. Jin, and C. Bystroff. Mining residue contacts in proteins using local structure predictions. In *IEEE International Symposium on Bioinformatics and Biomedical Engineering*, pages 168–175, November 2000.

Chapter 2

UNDERSTANDING HIGH DIMENSIONAL AND LARGE DATA SETS: SOME MATHEMATICAL CHALLENGES AND OPPORTUNITIES

Jagdish Chandra

Abstract Spectacular advances in sensor technology, data storage devices, and large-scale computing are producing huge data sets. These large and high-dimensional sets arise naturally in a variety of contexts such as the dynamics of the Internet, imaging for surveillance and diagnostics, and gene sequencing. The significant change in the scale and complexity embodied in these types of data, as well as the intricacies of the underlying phenomena being studied, present some new conceptual challenges. There has been considerable research activity dealing with the organization and analysis of such large data sets. But, by and large, these approaches have had only limited success towards the goal of understanding fully the inherent structures of these large data sets. There is a need, therefore, for new fundamental thinking about these problems and new mathematical approaches. In this paper we review a few such promising directions that draw extensively from fertile areas of harmonic analysis, discrete mathematics, stochastic analysis, and statistical methods.

Keywords: hyperspectral imagery, curse and blessing of dimensionality, exploratory data analysis, performance measures, multi-scale and multi-resolution analysis, and inverse problems

1. Introduction

It is now common knowledge that data gathering, data management, and data processing are routine practices in both the scientific and commercial world. Spectacular advances in sensor technology, data storage devices, and large-scale computing are enabling collection of huge data sets, perhaps terabytes of data, which tend to lie in very high dimensional spaces. Examples of these abound. For instance, we are all fascinated by the progress made in the gathering of data about the human genome. This is but just the beginning as the genome is only indirectly related to the much sought after and important protein function, and this information in turn leads to the next level

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 23–34.
© 2001 *Kluwer Academic Publishers.*

of understanding, namely, that of the overall cell function. In fact, it is not an exaggeration to say that we are witnessing the initial stages of a historic journey from genomics to proteomics and beyond. It is inevitable that at each stage, we are likely to collect more and more data describing various attributes, and analyze it to draw some conclusions.

Equally tantalizing are the data sets collected from satellite and hyperspectral imagery. In the first case, the goal is to compile a database that will map the entire surface of the earth to a very high resolution (say, 1 meter). The benefits of such exploration, to the discovery of natural resources and to agriculture, are enormous. This will be particularly significant to the developing countries throughout the world. In the second instant of hyperspectral imagery, multiple spectral analysis, for example using new optical sensors involving thousands of spectral bands, may reveal crucial information about the chemical composition of various components, thus providing enormous insights into crop identity and spread of diseases in crops. This will then be extremely important in providing a crucial understanding of the effects of pests and also of the dreadful scourge of draught around the globe.

Financial and consumer data, growing in leaps and bound, is being used increasingly in a critical way in web marketing, e-commerce, and in global financial markets. Gathering information about browsing and shopping behaviour of consumers is enabling the development of common preference data. Dynamics of the Internet, neural recording and other medical diagnostic data, and large-scale scientific and engineering computing provide many other challenging examples of large and high dimensional data. Where as data sets in one, two, and even three dimensions are easily visualized and analysed, data sets in higher dimensions (say, dimension one thousand) are much harder to understand. In order to understand large data in such high dimensions, one needs to overcome many technical challenges. Consider the problem of "overfitting" the data. For instance, even a trillion points are a very sparse set in a thousand dimensional space. It is easy to overfit data with a model that makes one suspect the existence of some pattern that may quickly disappear with the addition of more data.

A few more words about the significance and the role of data analysis to the society at large and the national security in particular. We are becoming a data/information driven society. Every aspect of our life depends on our ability to collect, analyse and understand large sets of data and information. This is nowhere as critical as in the battlefield. Data analysis in the battlefield of the future must be quick and accurate, efficient and robust. For example, the serial/parallel analysis of distributed data for detection and tracking of incoming missiles is critical. At any given time during the prosecution of a typical combat interaction, however, only part of the data will normally be required at the central decision point (Command Center). The needs of the Army in the near future, therefore, revolve around the efficient use of databases that are distributed in a network of nodes (combat units) that comprise the network (the friendly-force-in-battle). Each of the nodes retains data that are of local character and importance, some of which may be of value to other nodes (units) as the battle develops. The retention of all network data at one command site, however, may be counterproductive to the efficient use of data (in the conduct of the battle) because most of the data will not be used at the command node. When certain data are required, they must be accessible quickly, easily, reliably, and safely. It must be noted that such scenarios are not limited to military applications and are, in fact, the common characteristics of many other civilian applications. What is clear, therefore, that there is a need for distributed databases, a need for efficient and reliable data representation/compression schemes, and fast, safe and reliable data transmission protocols [C98].

Thus, a major challenge is to find methods from geometry and topology to analyse the structure of such sets, to fit models robustly, and identify and validate patterns. At present we do not have effective and efficient approximation schemes for such ever-increasing high-dimensional data. Classical methods are simply not adequate or designed to cope with this kind of explosive growth of dimensionality of observations. In the rest of this paper we will briefly describe how mathematical sciences will play a critical role in different phases of understanding data, in defining and finding patterns in such high dimensional data, and in the development of effective and efficient tools to retrieve, analyse and visualise data. We will evaluate the impact of both the much talked about "curse" of dimensionality and perhaps the less explored and very intriguing "blessings" of dimensionality.

2. Common Features of Data and Statistical Models

One of the key characteristics of current data sets is their size. A second important feature is the complex structure in terms of relations between different parts of the data and the nature of the data itself. Typical examples include two-dimensional images, three-dimensional objects as produced, for instance, by models of atmosphere, and four-dimensional temporal evolution of three-dimensional objects. Most of the measurement processes and observations encountered are noisy and are inherently subject to random fluctuations. Equally critical is our imperfect understanding of the basic processes that we may be after. In many situations we are forced to rely only on the indirect observations of what actually is desired. For instance, in genetics, we are able to observe only the gene correlations, and not their actual interactions. Thus, the complexity of structure, the noisiness of observations and the imperfect understanding of models, lead us naturally to the construction of robust statistical models for what is observed.

There are two principal aspects of the use of statistical models, namely, exploratory and confirmatory. In the exploratory phase, models are used to describe various features of the data, which may be important for understanding. For instance, hidden Markov models are usually fit to genetic sequences to identify common features of different sequences and patterns in a single sequence. In the confirmatory data analysis, one assigns probability to the features observed to ensure that what we are seeing is not due to chance. Both aspects are essential. From the point of view of display and visualization, this may also alert or bring to our attention facts about a specific feature, even before we are able to do anything about it. In the current data environment, models applied to huge and complex data sets may lead to all sorts of interesting findings, many of which may turn out to be spurious. The challenge is to use the tools of confirmatory data analysis to impart credence to such models.

Such questions are not new and have been around and in fact have stimulated the development of certain branches of mathematical techniques and probabilistic theory. One can even trace their origin to Gauss and Laplace. Some of the prominent contributors to these topics during the last century included Kolmogorov, Levy, and Feller. The founders of modern statistics like Fisher, Neyman, Rao, Wald and many others have made significant contributions to this field. But what requires rethinking of old ideas and infusion of new ideas is the dramatic increase in scale of problems as well as their internal complexity.

3. Early Stage of Modern Data Analysis

John Tukey [Tu77] was the pioneer who emphasized the exploratory nature of data analysis. Over the past forty years or so, under the influence of Tukey and many others, tools and procedures have been developed at an unprecedented pace for the analysis and representation of data. These techniques have contributed to the rapid advances of information technology in areas such as massive data storage, efficient retrieval algorithms, rapid throughput, and interactive display and visualization. Software packages such as S, S-plus, CART, Data Desk, GLIM, MacSpin, and statistical procedures like bootstrap [E82], and Monte-Carlo Markov chain (MCMC) [DS95] have made a major impact on modern data analysis. However, the growing nature of high dimensional data sets with various levels of uncertainty (both at model/structural level as well as at the observation level), encountered routinely in various application domains including information technology, have substantially stressed the limits of these methods. New mathematical insights are badly needed that particularly would grapple with the curse of dimensionality and that would also fully exploit the potential of the blessings of high dimensionality [D00].

The curse of dimensionality, so eloquently introduced by Richard Bellman [B61], generally refers to the intractability of systematically searching through high dimensional spaces. Function analytical methods from harmonic analysis such as the wavelet transforms [SN96] have been very successful in overcoming the curse of dimensionality. Multi-scale and multi-resolution methods (particularly the wavelet-based methods) provide powerful tools in high dimensional data analysis. Development and utilization of these methods have given a major boost to many fields of applications, as if furnishing the performers in these domains with a microscope to analyse the data. There are several useful linkages between wavelet transform analysis and multi-dimensional data analysis [Mu98]. These methods can be fast and effective. Effective software packages have been developed that combine the wavelet transform procedures for data representation and analysis, data visualization techniques, and data matrix manipulation High performance software based on such innovative approaches includes MR/1 and MR/2 [SMB98]. For visualization of scientific and engineering data, these packages include linear and B-splines based wavelet transforms, wavelet transforms due to Haar and Mallat, pyramidal and redundant wavelet transforms, and a range of non-wavelet multi-resolution transforms such as morphological median, morphological min-max, and various other methods. Among the noise modeling, these packages support Gaussian and Poisson models, non-stationary additive noise, stationary and non-stationary multiplicative noise, and a locally adaptive strategy for undefined noise.

The blessing of the dimensionality, on the other hand, accrue from the geometric properties of abstract Banach spaces such as concentration of measures. The idea of concentration of measure was introduced by Vitali Milman [Mi88] (inspired by the work of Paul Levy) and other related work on asymptotic limits as the dimensionality increases. This usually implies that certain random fluctuations are very well controlled in high dimensional spaces, lending themselves to effective asymptotic methods, which otherwise would appear to be more complicated in moderate dimensions. An interesting survey of some of these ideas can be found in [Ta96]. This area is not fully explored in the context of understanding high dimensional data sets. For an excellent exposition of these possibilities, we refer to [D00].

4. Multi-Scale, Multi-Resolution Analysis

Among many data intensive applications, multi-scale analysis and wavelet theory have had their largest impact on information processing, including coding, extraction, enhancement, and distribution. This is because of their several advantages:

- Wavelet bases have local support in both space/time and frequency. This allows capturing features in both domains of signal processing.
- Wavelets provide unconditional bases for function spaces. This leads to the derivation of optimal numerical algorithms by solving functional and analytical extremal problems in data/image processing.
- Hierarchical structures provided by the multi-resolution methods provide the ability to decompose data/images from coarse to fine resolution/details, and from low to high frequency. This is significant for feature classification, object recognition and data mining since the desired details can often be identified with certain scales.
- This single technology offers a robust and versatile approach to handle all different tasks of information and data processing in an integrated platform.

Given the impressive achievements of multi-resolution methods, there is still considerable possibility for increased performance by integrating highly nonlinear analysis and novel statistical methods. Specifically, the emphasis here should be on analysing the efficiency of highly nonlinear methods to derive complementary theories and algorithms. It is clear that progress here can benefit a variety of civilian and military applications. These include rapid data query and navigation, compression and feature extraction for telemedicine, automatic target recognition, autonomous navigation for military and civilian use, the detection of micro-calcification and lesions in digital mammography, the detection of cracks and other abnormalities in aircrafts, and the registration of digitized images. For example, compression is vital for data transmission and rapid access of multi-media information. To enable telemedicine on the battlefield, compression must be utilized to accelerate communication and as a pre-processor to feature extraction. Compression is normally achieved by quantization. This implies that only certain significant bits of information are retained. For transmission or storage, the quantized coefficients are encoded. Various encoders and compression algorithms that utilize the spatial correlation of wavelet coefficients in a quad tree data structure and develop special encoders for the ensuing tree have been designed. However, management of information using adaptive approaches and smart data structures still need to be better understood before the potential of novel basis functions can be fully realized.

Conventional maps are highly abstracted representation of information derived from imagery. The process of extracting salient features is now largely accomplished by manual inspection. Manipulation of this type of imagery and other data, such as automatic extraction of important features in addition to lossy data/image compression, archiving, and electronic distribution of resulting product, will be needed. One of the most common methods in feature extraction is to find an efficient representation of the data/image in terms of a highly redundant dictionary of functions. As such, there is a great need and interest in the development of novel and efficient algorithms that achieve near-optimal approximation. Multi-resolution approaches combined with geometric computing provide powerful tools for solving a variety of problems such as adaptive network routing, compact representation and manipulation of geometric databases, texture mapping, surveillance and reconnaissance, hypermedia processing, and visualization.

5. Data Models and Performance Measures

Genuine multi-disciplinary efforts are needed for developing reliable frameworks for understanding the evolving structure of large data sets. At a fundamental level, stochastic models have to be studied for very complex spaces of objects, trees and other combinatorial structures, images, and multi-dimensional objects. Non-parametric classification and regression methods had some success in applications such as speech recognition and machine learning. Complicated classification algorithms are usually developed on the basis of a training set of data. For instance, computer speech recognition algorithms are trained on the basis of large samples of speech of one or more speakers. These algorithms may be based on combination of hidden Markov models, neural networks, quantization and other procedures. The basic question is how one can predict, in advance, the performance of these algorithms on a new set of speakers. Measuring performance on the training set can be highly deceptive, since the available number of parameters of the algorithm may be such that one could, in principle, recognize them almost perfectly. But what happens when we are faced with new samples? This is what we alluded to earlier as the danger of overfitting the data. Such issues can become even more acute in other applications such as pattern recognition or hunting for functionally important sites in protein. More general problems of learning and optimal decision making in uncertain environments offer many technical challenges.

Good measures of performance require probabilistic analyses of the algorithms. These are important not only for realistic performance measurement but also for qualitative understanding of many types of procedures. Some interesting and promising work in this direction includes bounds and minmax results [V00]. But unfortunately for many applications these bounds tend to be unrealistic and provide only limited qualitative understanding. Intensive efforts are needed here that could immensely benefit a broad class of applications ranging from information fusion to learning and adaptation.

Computer models are being developed and frequently arise in many fields ranging from image analysis and atmospheric models, to complex interactive networks and systems that describe models of transportation, communication, and electric power systems. The models may be deterministic, mixed, or stochastic. These may involve a large number of partial differential equations with inputs or boundary conditions involving noisy data, algebraic constraints, and hybrid systems. Validation of these models continues to be a major challenge. The scale of some of these models which may run for many days on the most powerful computers, forces one to make many compromises. For instance, one may be faced with how resources are to be allocated between replication of experiments, construction of better algorithms, and on building in improvements in the underlying theory in refining the scale of models. In order to make these compromises intelligently, one needs to study the intersections of statistical uncertainties coming from data, with the numerical errors incurred, with the model uncertainties, with computational speed and resources, and all this in the context of what the model is going to be used for. These problems therefore offer important challenges for fruitful collaborative work among applied mathematicians, numerical analysts, statisticians, and computer scientists.

6. Discrete Mathematics

Discrete mathematics serves as a bridge that links mathematics to communication and computing. This is the bedrock of the digital (information) revolution. For example, spectral methods from harmonic analysis are increasingly used in graph algorithms for dealing with massive data sets. The last century witnessed a large part of traditional mathematics being heavily motivated by physical sciences. With information technology and biological sciences as the driving forces, new challenges await discrete mathematics. Consider the example of graph embeddings and massive graphs. The basic problem here is to embed one graph into another, subject to constraints such as preserving distances and other invariants. This is to be accomplished in an efficient way, by minimizing the cost. Of particular interest in relation to large data sets are the fundamental relations between a graph and its sub-graphs. The relevant topics here include containment, avoidance, and partitioning. For example, visualization and representation of massive data sets can be viewed as projection of a large graph into a small chosen graph. When only partial information about the graphs is available or sizes of the graphs are prohibitively large, many new and interesting research problems emerge. In another direction, the inverse problem of constructing a graph from its projections has important applications in memory management, computational biology, and Internet tomography (see the next section).

7. Model Rich Data Mining (Inverse Problems)

We have witnessed during the last century a greater reliance on mathematical models to describe key processes in science, engineering, technology, and business. While the pace of increase of such models was vast, the reliance on mathematical models is not new. This goes back to Kepler's mathematical description of the motion of planets and to Newton's law of gravitational forces. What is exciting is that we are likely to see many more instances of these in the coming decades from biology, social sciences, and finance. The key issue in many of these models is that while the general structure of the model may be well understood, this may not be the case with specific features or characteristic numbers associated with the process in question. For example, one may observe the structural changes in the response of material due to cracks and flaws to identify the actual location of the flaws. One can measure the conductivity or density of minerals or ores in the Earth's mantle, in order to identify the location of these minerals. One can observe the change in the stiffness in human tissue perhaps due to cancerous growth, and then be able to identify the location of those growths. Similar examples are encountered in financial markets and information technology (see the example on the Internet, to follow).

In order to calibrate mathematical models and render them useful to the users of any discipline or application field, there is a need to efficiently extract the hidden parameters from the observations via the models themselves. In inverse methods in mathematics, models are used "backwards", to infer their underlying parameters. This is accomplished using the data from the measurements or observations, rather than predicting the measurements from the parameters. Going back to our examples, in Newton's law, the speed would not be found knowing the gravitational acceleration constant, but rather the gravitational acceleration constant is determined from the measurement of velocity.

In these " model rich" data mining methods, large data sets are analysed with the aid of mathematical models. In these situations, it is not enough to have merely computational power. We need a deep understanding of the nature and mathematical structure underlying this massive data. Unlike the popular perception about the digital revolution, the rapid advances in brute force computing capabilities alone cannot achieve the potential of information technology and its beneficial implications to the society at-large.

Many of these generic model rich data mining problems have some important common characteristics. These include dependence on very large and information rich data sets, intrinsic uncertainty both in model/structure and data, and inherently nonlinear behaviour. We will conclude this paper by describing an example that encapsulates many of these characteristics.

8. Example (The Internet)

The Internet is a vast and rapidly changing collection of connections (edges) and computers (nodes). In fact it has become so large and dynamic that an attempt to provide a detailed description of it as a network, in a rather traditional mathematical sense, becomes very unwieldy and of very little practical value. However, if one looks at a coarser spatial scale, perhaps, measured in connectivity (and not distance), and if one observes it over only short time windows, the Internet exhibits statistically stable behavior. This apparent stability opens up the possibility of a description of network traffic that does not depend on the detailed tracking of each and every packet.

Instead, we can divide the network into regions and seek the rate of packets passing between contiguous regions. Assuming a statistical model (for example, independent Poisson process) for each source- destination pair, it is possible to estimate the parameters of the process from either data on contiguous-region traffic recorded by routers, or even from data on end-to-end traffic for only a few of the myriad source-destination pairs. This is a typical problem of inverse type, commonly referred to as Internet tomography.

Having estimated the statistics of contiguous region traffic, it is possible to reconstruct the statistics of the traffic between any two regions within the Internet, and hence to obtain economically valuable information about who is messaging whom, and what changes in traffic result from changes in connectivity. Clearly, a successful probabilistic model will make the management of the Internet and its likely rapid growth more tractable. With the economic potential of this activity soaring past a trillion dollars in near future, it has captivated the attention of many. What is called for is a multi-disciplinary effort that involves applied mathematicians, statisticians, signal processing engineers, communication, and computer scientists. There are many technical challenges. These include development of sound methods for sub-dividing the network to maximize the effectiveness of the inverse methods. We need a better understanding of what statistical models are appropriate for contiguous region traffic. Finally, there is a need for the development of algorithms to estimate the parameters of fairly comprehensive statistical models of traffic.

9. Conclusion

The basic goals in dealing with large and high-dimensional data are usually to seek proper representation/ compression, data transcription for various purposes, and perceptual communication with the data. The existing approaches in data analysis are becoming inadequate as each year new sensors and other sources of data collection are invented. We are becoming a data/information driven society. But the well-established and tested techniques are being pushed to their limits of applicability. For instance, the common assumptions of Gaussianity and linearity in data and associated models are incapable of representing realistic situations. Therefore, many of these high-dimensional data analysis problems require new and innovative mathematical approaches and statistical procedures. On a wider scale, the program of digital libraries, genome project, and digitized battle space provides a growing number of convincing examples to show our ability to collect and store all data/ information surrounding us. The key issue, however, is how this data/information is made accessible, understandable, and useful. Central to this task is the question of proper representation of data/information. Promising content and context-based methods will require highly synergistic efforts involving mathematical, computer, information, behavioural, and social scientists.

Acknowledgements

The author wishes to thank the referees for their many valuable comments and suggestions. He also acknowledges the partial support from DOD/EPRI under the initiative on Complex Interactive Networks and systems.

References

[B61] R. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, 1961.

[C98] J. Chandra, Integration of data fusion and network control, *Proc. Workshop on Data Compression Processing Techniques for Missile Guidance Data Links*, (Keynote Address), 411-440, 1998.

[DS95] P. Diaconis and, L. Saloff-Coste, What do we know about Metropolis algorithms, *Proc. 27th Annual ACM Symposium on Theory of Computing*, 112-129, 1995.

[D00] D. Donoho, High-dimensional data analysis, *Amer. Math. Soc. Symposium on Mathematical Challenges of 21st Century*, August 2000.

[E82] B. Effron, *Jackknife, Bootstrap, and other Resampling Plans*, SIAM Publications, 1982.

[Ma98] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

[Mi88] V. Milman, The heritage of P. Levy in geometrical functional analysis, *Asterisque,* 157/158, 273-301, 1988.

[Mu98] F. Murtagh, Wedding the wavelet transform and multivariate data analysis, *Jour. Classification*, 15, 161-183, 1998.

[SMB98] J. Starck, F. Murtagh, and A. Bijaoui, *Image and Data Analysis: The Multi-scale Approach*, Cambridge University Press, 1998.

[SN96] G. Strang, and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, 1996.

[Ta96] M. Talagrand, A new look at independence, *Annals of Probability,* 24, 1-34, 1996.

[Tu77] J. Tukey, *Exploratory Data Analysis*, Addison Wesley, 1977.

[V00] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 2000.

Chapter 3

DATA MINING AT THE INTERFACE OF COMPUTER SCIENCE AND STATISTICS

Padhraic Smyth

Abstract This chapter is written for computer scientists, engineers, mathematicians, and scientists who wish to gain a better understanding of the role of statistical thinking in modern data mining. Data mining has attracted considerable attention both in the research and commercial arenas in recent years, involving the application of a variety of techniques from both computer science and statistics. The chapter discusses how computer scientists and statisticians approach data from different but complementary viewpoints and highlights the fundamental differences between statistical and computational views of data mining. In doing so we review the historical importance of statistical contributions to machine learning and data mining, including neural networks, graphical models, and flexible predictive modeling. The primary conclusion is that closer integration of computational methods with statistical thinking is likely to become increasingly important in data mining applications.

Keywords: Data mining, statistics, pattern recognition, transaction data, correlation.

1. Introduction

The goal of this chapter is to explore the past, present, and potential future relationship of statistics to data mining, and to further argue that statistics should play a foundational role in any data mining endeavor. The target audience of this chapter is intended to be computer scientists, engineers, and scientists who are knowledgeable about the algorithmic aspects of data mining but who would like to learn more about how statistical ideas and concepts may help them in their work. Hopefully this chapter will provide at least a few pointers for such readers.

The term "data mining" has been used in a variety of contexts in data analysis in recent years. As a reader of this book it is likely that

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 35–61.
© 2001 *Kluwer Academic Publishers.*

you already think of data mining from the computer science viewpoint, namely, as a broad set of techniques and algorithms for extracting useful patterns and models from very large data sets. Since the early 1990's there has been a broad surge of both research and commmercial activity in this area, largely driven by computationally-efficient massive search for patterns in data (such as association rules) and by business applications such as analysis of very large transactional data archives. As evidenced by well-known business-oriented data mining texts (e.g., [BL00]), as well as various published proceedings of data mining research conferences, much current work in data mining is focused on algorithmic issues such as computational efficiency and on data engineering issues such as data representation. While these are important topics in their own right, statistical considerations and analyses are often conspicuous by their absence in data mining texts and papers. This is a natural consequence of the fact that research in data mining is largely practiced by computer scientists who naturally focus on algorithmic and computational issues rather than on statistical issues. In this chapter we explore the interplay of computer science and statistics and how this has impacted, and will impact, the development of data mining.

2. Is Data Mining Different from Statistics?

Is data mining as currently practiced substantially different from conventional applied statistics? Certainly if one looks at the published commercial applications of data mining, such as the case studies presented in [BL00], one sees a heavy reliance on techniques that have their lineage in applied statistics. For example, decision trees are perhaps the single most widely-used modeling technique in commercial predictive data mining applications [Joh99, Koh00]. They are particularly popular because of their ability to both deal with heterogenous data types (they can easily handle both categorical and real-valued variables) and to find relatively low-dimensional parsimonious predictors for high-dimensional problems. Many other techniques that are popular in data mining also have their roots in applied statistics, such as nearest neighbor models, naive Bayes models, and logistic regression for prediction, and k-means and mixture models (using Expectation-Maximization (EM)) for clustering and segmentation. Arguably it is association rules that are the only main exception, i.e., association rules are a technique that have no clear "ancestor" in the statistical literature. It is also debatable as to how many successful real-world applications in data mining actually rely on association rules for their success.

Thus, while the annual proceedings of conferences such as SIGKDD and SIGMOD contain many novel and interesting techniques that are not within the mainstream of conventional statistics, when one looks at both current data mining software tools (from the likes of IBM, SAS, SGI, and many others) and at industry-specific applications (such as domain-specific applications developed in e-commerce) there is a heavy reliance on the application of traditional statistical ideas. Indeed, a statistician might argue that data mining is not much more than the scaling up of conventional statistical methods to massive data sets, in effect a large-scale "data engineering" effort.

While there is some truth to this viewpoint, a more accurate reflection of the state of affairs is that data mining (and more generally, computer science) has indeed introduced a number of new ideas within the general realm of data analysis, ideas that are quite novel and distinct from any prior work in statistics. We can identify several such contributions that have arisen primarily from work within computer science (data mining, machine learning, neural networks) rather than from conventional statistics:

1 **Flexible predictive modeling methods**: from the early work on decision trees and neural networks to more recent techniques for combining models, there is a history of computer scientists introducing many new techniques for predictive modeling, particularly for classification problems. These techniques often start out with strong algorithmic foundations but weaker formal statistical justification. Over the past 20 years in machine learning and data mining there has been a recurring pattern of flexible models and algorithms being introduced, developed, and applied by computer scientists (e.g., the boosting framework of Freund and Schapire [FS97]), with the supporting statistical theory being "filled-in" at a later date (e.g., the statistical justification of boosting provided by Friedman, Hastie, and Tibshirani [FHT00]).

2 The use of **hidden variable models** for large-scale clustering and prediction problems. For example, hidden Markov models are an excellent example of how complex non-stationary data (a speech signal for example) can be "compressed" into a relatively simple hidden state representation that is often quite adequate for classification and prediction applications (see [Ben99] for a review). The EM framework for training hidden variable models such as HMMs is appealing to computer scientists since it is couched as an algorithm, and its algorithmic basis has led to its widespread

adoption by computer scientists in a variety of applications (e.g., for model-based collaborative filtering, Heckerman et al., 2000).

3 **Finding patterns** (data mining) rather than global models (statistics): examples of pattern-finding algorithms include association rule algorithms [AIS93], sequential association algorithms [MTI95], rule induction algorithms [WB86, SG92, FF99], and contrast set algorithms [BP99]. These pattern-finding algorithms differ from more conventional statistical modeling in that they do not attempt to "cover" all of the observed data, but instead focus in a data-driven manner on "local" pockets of information.

4 **The engineering of scale**, namely, the data engineering aspects of scaling traditional algorithms to handle massive data sets. Work in this area involves both computationally-driven approaches [ZRL97, ML98, BPR98, GGRL99, PK99] as well as statistically-motivated techniques [DVJ99]. Worth mentioning in this context is the fact that researchers in a variety of areas such as speech, natural language modeling, human genome analysis, and so forth, have all developed a variety of practical learning algorithms and "tricks of the trade" for dealing with massive data sets, e.g., [LB97, WMB99].

5 Analyzing **heterogenous structured data** such as multimedia data (images, audio, video) and Web and text documents. For example, there is a significant research within computer science on using learning algorithms to improve our understanding of the structure of the Web (e.g., [Kle98]) and in learning how topic categories can be automatically extracted from documents (e.g., [Hof99]).

It is important to emphasize that statistics still plays a major role in each of these areas. However, to a large extent it is not statisticians who are leading the charge but rather computer scientists and engineers who are adapting statistical methods to these new problems and opportunities.

There are other general distinctions between data mining and statistics. For example, data mining is typically (indeed, in practice almost always) concerned with observational retrospective data, i.e., data that has already been collected, often for some other purpose (e.g., records of financial transactions recorded by a bank for accounting purposes). Thus, issues such as experimental design (the construction of an experiment to collect data to test a specific hypothesis) are not typically within the vocabulary or tool-set of a data miner. For other general discussions on statistical aspects of data mining see [EP96, GMPS96, GMPS97, HPS97, Han98, Lam00, Smy00].

3. A Reductionist View of Data Mining

Let us consider a very high-level view of data mining and try to reduce a generic data mining algorithm into its component parts. The particular reductionist viewpoint proposed here is not necessarily unique, but it nonetheless does provide some insight into the different and relatively independent components that make up data mining algorithms. As discussed here this breakdown is focused on *algorithms*, rather than the overall data mining *process*: the overall process involves numerous additional (and important) steps such as data selection, data preprocessing, evaluation, and so forth, which we will not discuss in any detail.

In this proposed framework, we identify five primary components of a data mining algorithm (see [HMS01] for further discussion):

1 **The Task:** In data mining we can consider our tasks to fall into a few general categories: exploratory data analysis (e.g., visualization of the data), pattern search (e.g., association rule discovery), descriptive modeling (e.g., clustering or density estimation), and predictive modeling (e.g., classification or regression). This is not perfect but it gives us a high-level categorization of common data mining tasks. An important point is that the task should (if possible) be identified explicitly in any data mining endeavor, since the nature of the task will directly influence the other components described below.

2 **The Model Structure:** This is the specification of the structural form that we seek in the data. For descriptive tasks (e.g., clustering) the structure may be of paramount importance, while in predictive tasks the exact nature of the structure may be only of secondary importance compared to the predictive power of the model. Examples of general structures include decision trees, Gaussian mixtures, association rules, and linear regression models. For example, for a regression problem with 2 input variables X_1 and X_2 and a target variable Y we might propose as a simple linear relationship:

$$Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + e \tag{3.1}$$

where e is random additive noise (e.g., Gaussian, zero mean). In this case our model structure is a plane in the two-dimensional X_1, X_2 space with noisy observations (Y values) distributed about this plane. A *specific* model consists of the specification of a particular plane parametrized by $\alpha_0, \alpha_1, \alpha_3$, the three unknown parameters of the structure. Once we specify a particular structural form we effectively limit (for better or worse) our view of the data to a view that is relative to that structure. Naturally, in practice, it is common to specify multiple different types of structures (e.g., both decision tree and logistic regression models for a classification problem) so as to allow the data to indicate which structure is best. We do not need to assume that the true data-generating mechanism is within our model family. Usually our model structures represent convenient approximations to some unknown reality.

3 **The Score Function:** Loosely speaking this is how we evaluate the quality of a particular fitted model or pattern on the basis of observed data, e.g., squared error for regression, classification error for classification, and so forth[1]. The score function provides a link between the hypothesized model structure and the observed data. Naturally, the form of the score function is highly dependent on the nature of the data mining task being undertaken. It is typically defined as a function of the unknown parameters of the model given a particular observed data set. In practice it is customary to choose a score function from a relatively small set of well-known and easily analyzed functional forms, such as squared error for prediction of real-valued quantities, e.g.,

$$S(\alpha_0, \alpha_1, \alpha_2) = \frac{1}{N} \sum_{i=1}^{N} \left(y^i - (\alpha_0 + \alpha_1 x_1^i + \alpha_2 x_2^i) \right)^2 \qquad (3.2)$$

where (y^i, x_1^i, x_2^i) is the ith training tuple, $1 \leq i \leq N$. Thus, we view our score function S as a one-dimensional function of the three unknown α parameters and seek to minimize S given the observed data. In fact we are free to choose a completely general score function to reflect the actual losses (or gains) that would be incurred in practice when using the model, e.g., a general loss matrix for classification rather than simply counting errors. Note that the score function may be used in different ways, e.g., evaluated both on the training data (to search for good parameters) and on a validation or test data set (to search for a good model from a set of fitted potential models, each of which had its parameters fitted on a training data set).

4 **The Optimization Algorithm:** Given a score function, a model structure, and an observed data set, the problem of defining a data mining algorithm can now be reduced to that of defining a computational procedure for finding the extremum of the score function

(e.g., minimizing the squared error) over parameter space (and over model space when model complexity is allowed to vary or multiple model forms are present). In a few cases the problem can be solved in closed form. For example, finding the α's that minimize S for the regression problem above can be solved using linear algebra. Similarly, determining the probabilities used in a naive Bayes classifier can be carried out by simply counting relative frequencies of different events. More typical, however, is the case where the score function has a complicated surface as a function of the parameters, with multiple extrema (keep in mind that if we have k parameters then the score function is a scalar function defined over a k-dimensional space). In such cases finding the global extremum of the score function may be an NP-hard problem. Thus, we must resort to some form of heuristic search, e.g., greedy search among the space of all possible decision trees, gradient descent with multiple random restarts in the space of neural network weights, and so forth. This optimization/search component is the heart of the actual data mining algorithm. However, it is important to be aware that a clever optimization algorithm for fitting a model is really only as useful as the model structure, score function, and task that it is based on. In other words, attention needs to be paid that we are in fact asking the right questions of the data (see [Han94] for further discussion in a statistical context).

5 **The Data Management Strategy:** Given an actual algorithm, the final step concerns the implementation of specific mechanisms for how the data are accessed, stored, and managed. In many research papers no explicit data management strategy is specified. For example, most existing machine learning and statistical learning algorithms just assume that the data are read into main memory and reside in a flat file. Data miners are keenly aware that data management can be extremely important in practical applications involving very large data sets. Algorithms such as the A Priori algorithm for association rules make the data management component very explicit in the description of the overall algorithm (i.e., multiple linear scans of the data). Other work takes an existing framework where the model/score function/optimization components have been well-defined in the literature and are tried and tested in practice, and then embeds them in an explicit data management framework where efficiency of data access is given priority. Examples of this approach include the BOAT framework for efficient decision tree fitting to massive data sets [GGRL99], and the

general ADTree data structures of Moore and colleagues for efficient data access and caching for a variety of learning algorithms [ML98, Moo99].

The five-component framework above provides us with a simple systematic language for understanding the "parts" that make up a data mining algorithm. Typically the modeling is guided by concepts from applied mathematics and probability, the score function is often driven by general principles of statistical inference (i.e., how we want to connect an abstract model to actual observed data), and the optimization and data management components are heavily algorithmic in nature. By describing a data mining algorithm within this framework we can understand somewhat more clearly what the primary contribution of each component of a specific algorithm is as well as providing a reference framework for systematically comparing different algorithms and synthesizing new ones.

As an example, data mining algorithms based on either regression trees or neural networks differ primarily in terms of their model structure (hierarchical piecewise constant functions for trees, versus non-linear combinations of weighted sums for networks), as well as in terms of how the optimization is carried out (greedy "one variable at a time" structure search for trees, versus gradient descent in continuous weight-space for networks).

Similarly a general technique such as genetic algorithms is best viewed as belonging to the optimization component of a data mining algorithm, rather than being viewed as an algorithm or model representation unto itself. The nature of the genetic optimization framework imposes representational restrictions on the model structure being used (the model representation must typically be defined as bit-strings in some form). But the primary role of a genetic algorithm is to provide a non-local search strategy in model-space, in contrast to the more traditional local (e.g., gradient-based) search methods. By decoupling the model structure component from the optimization/search component we can mix and match different model structures and optimization techniques in any manner we wish.

Within this framework, we can see that data miners typically place much more emphasis than statisticians on the two primary computational components (optimization and data management). Conversely, statisticians typically place much more emphasis on the model representation and score function components (modeling and inference) than on the computational aspects of how the model is fit. To see examples of this one need only look at typical research papers from each field (e.g., from the *Journal of Data Mining and Knowledge Discovery* and the *Journal of the American Statistical Association*). While this represents two extreme ends of a spectrum (computational issues at one end, modeling/inference at the other) it is nonetheless a useful spectrum along which to consider how data mining and statistics differ. Each community is comfortable with what they know best: computer scientists grow up on a basic diet of algorithms and data structures, while statisticians grow up on a basic diet of mathematical models and principles of statistical inference. Both aspects can play an important role in data mining.

4. An Example: Modeling Transaction Data

Let us briefly consider the problem of analyzing transaction (market-basket) data within the general reductionist framework outlined in the previous section. Association rules are a widely-used technique in data mining for analyzing such data, and are primarily intended for the task of pattern discovery (although one can also consider secondary uses of association rules, such as clustering and prediction). The underlying association rule "structure" is a set of rules, statements of the form "IF A and B and C THEN D with probability 0.9" where A, B, C, and D are all binary propositions (true or false). The emphasis in association rule algorithms is not so much on the nature of the task or the model representation, but more on the search and data management aspects of the algorithm. The essence of the idea is to use fast counting and breadth-first systematic search to find the set of association rules that are above certain pre-defined constrains on support, $p(A, B, C, D)$, and confidence, $p(D|A, B, C)$. Papers on association rule algorithms tend to emphasize computational efficiency of the algorithm itself (the efficiency of the search and data management component) rather than interpretation or evaluation of the rule sets that are obtained.

Association rule algorithms are a good reflection of the database-oriented view of data mining, where the problem of finding all association rules satisfying given confidence and support thresholds can be viewed as a general form of a database query. This contrasts sharply with the statistical view, which places more emphasis on the question of "what to compute," whereas the database researcher places emphasis on the question of "how to compute." For very large data sets in particular, both viewpoints can be thought of as equally valid and important.

As an example, for transaction data, a statistical modeller might propose a probabilistic model for how purchases are made. The arrivals of an individual shopper to a store could be modeled as coming from a Poisson distribution (a constant arrival rate λ over time), where each shopper might have their own individual Poisson rate λ_i. As an extension one might perhaps allow the λ_i's to vary seasonally and/or in a non-stationary manner. Conditioned on the event of having arrived at the store, a shopper's purchase choice of n specific items from the k possible items could be modeled as n independent multinomial (k-way) trials, where n is sampled from a distribution on how many items are in a basket. The multinomials and market-basket size distributions could be modeled as being different for each individual, but where the distribution of parameters across individuals is constrained in some manner.

For small numbers of items in a specific category (e.g., modeling individual purchasing patterns of a particular brand of coffee) this type of Poisson-multinomial model has in fact been well-studied in the marketing literature (e.g., see [GEC84] for a seminal paper on this topic, and [WK98] for a comprehensive recent review). While such models inevitably are built conditioned on various assumptions (such as an assumed independence of the items being purchased from one store visit to the next), this "generative modeling" approach nonetheless can be quite powerful. It provides a systematic framework for hypothesis testing, clustering and segmentation, customer profiling, and forecasting (again see [WK98] for numerous examples of such models in modeling of consumer behavior, [MF01] for an example in the context of e-commerce, and [CGS00] for a general discussion of such generative models for clustering individuals).

In contrast to the association rule framework, this statistical approach to modeling of transaction data focuses strongly on the construction of a suitable data-generating model, with less emphasis on the computational aspects of how the model is fit to the data. Another difference between the two approaches is the fact that association rules look for *patterns* in the data, whereas the statistical model tries to construct a *global model* for the whole data set. Searching for patterns may be advantageous if that is the primary goal and if constructing a global model is difficult or unreasonable.

The main point in comparing the statistical and association rule approach to transaction data is that statistical models provide a flexible and general language for modeling such data, allowing aspects such as individual-level variations, temporal dependencies, and so forth, to be incorporated in a natural and systematic manner. These statistical techniques can be viewed as providing a useful and complementary set of tools to the more computationally-motivated data mining techniques such as association rules. For example, the statistical approach is based on certain modeling assumptions that may not hold in practice. In contrast, association rules are relatively "non-parametric" in the sense that there are few or no constraints assumed about the functional form of the data-generating process.

5. The Role of Statistical Education in Data Mining

In order to appreciate the fundamental role of statistics in data analysis, a data miner needs at least some minimal exposure to statistical concepts. Rather than learning a set of specific detailed models it is probably more important to appreciate the general mindset of statistical thinking, such as the explicit recognition of uncertainty in a broad data analysis context.

Many computer scientists and engineers engaged in data mining have had only a limited exposure to statistical ideas in their undergraduate curriculum (although engineers often receive more statistical education than computer scientists in most countries). While the value of statistical concepts are widely recognized, modern engineering and computer science curricula tend to be very crowded, leaving little room for anything other than a cursory introduction to statistics. For many graduates their only exposure to statistics consists largely of a cookbook-style class on how to use hypothesis tests (see [Lam00] for further elaboration on this topic).

There are at least three fundamental and important ingredients that are often missing from the statistical education of a typical data miner:

1 **Modeling Languages:** The first ingredient is actually not statistics at all, but rather mathematical and probabilistic modeling, languages for constructing explicit models for how the observed data might be generated. For example in our modeling of transaction data earlier, we proposed a constant arrival rate for shopper arrivals (modeled as a Poisson distribution) and a multinomial choice model for item choice. These are the types of basic building blocks that allow a statistical modeller to construct a relatively complex model from simpler functional forms. In terms of our reductionist viewpoint, this mathematical modeling provides the basis for determining the *model structure* component of an algorithm. The models may of course be informed by prior knowledge about the problem. A wide-ranging and widely used standard introductory text for this form of stochastic modeling is [Ros00]. [ED96] also provide a very readable introduction to a broad range of modeling methods in applied multivariate statistics, a topic of particular relevance to data mining. These are just two of many excellent texts on modeling: for example, there are many other more

specialized texts for specific classes of models, such as time-series models and spatial models.

2 **Inference from Data:** The second primary ingredient provides the foundations of inference: how to reason from the observed data to make inferences about the hypothesized model structures. In our reductionist framework, inference tells us (a) how to construct specific score functions from general statistical and decision-theoretic principles, (b) what the general and desirable properties of these score functions are, and (c) what these score functions can then tell us about generalization to new data. The score functions connect the hypothesized model to the observed data. To understand the principles of inference requires a foundation in the basic principles of mathematical statistics: expectation, parameter estimation, likelihood, sufficient statistics, and so forth. While it can be argued that an excessive emphasis on mathematical formalisms may not be productive in the context of the practicalities of data mining, these concepts and methods are essential if a data miner is to be equipped to penetrate the dense jungles of research publications and texts in statistics. In other words, some minimal knowledge of the basic concepts is required so that a data miner can access the many rich and foundational ideas on data analysis that the statistical community has provided down through the ages (and continues to provide). For readers interested in embarking on this journey, [Kni00] provides and up-to-date and readable introduction to mathematical statistics. Other well-known texts that are widely used in academic courses on mathematical statistics include [BD77, HC78, CB90].

3 **The Practical Art of Statistics:** While the first two educational components provide the raw mathematical tools of statistics, the third component provides the *art*, i.e., the art and craft of knowing how the analytical and modeling tools should be applied in practice. This is a much more subtle issue. Indeed it is fair to say that statisticians themselves are often not exposed to this "art" during their undergraduate education but tend instead to acquire it through years of experience. There are different levels to this "art." One level is the ability to know how to construct analytical models for a problem, e.g., the problem of modeling the time-dependent rate of arrival of customers at an e-commerce site or predicting the rate at which customers buy a given product as a function of demographics. Typically this is done by having access to a "toolbox" of basic functional forms and models. In the

e-commerce case we might model the "probability of buying" by some form of logistic regression, i.e., where

$$\log \frac{p(buy|X_1,\ldots,X_d)}{1-p(buy|X_1,\ldots,X_d)} = \alpha_o + \sum_{i=1}^{d} \alpha_i X_i, \qquad (3.3)$$

where the X_i's are different demographic variables such as age, income, etc., and the αs are again parameters of this logistic model that are estimated from the available data.

It is fair to say that of the three basic educational ingredients above, the most difficult one to learn is (not surprisingly) the "art." For example, the ability to successfully build complex models from basic functional forms is deceptively simple, yet it is often glossed over in the statistical literature, which provide little guidance to the uninitiated as to why certain types of models are preferred over others for a given application (with some notable exceptions such as [CS81, Cha95]). It is important to realize that much of statistical analysis is predicated on the use of an assumed model for how the data are generated. Even in non-parametric analyses (i.e., modeling techniques such as trees which rely on very few functional assumptions) there may nonetheless be various implicit assumptions present, such as assuming that the order in which individual observations are presented is irrelevant from the point of view of the model.

Another significant aspect of the "art" is the ability to dissect a problem to obtain insight into why a particular method works well or how it could be improved. Again, the techniques used are often relatively straightforward, but it is often not so obvious to know which technique to apply in which situation. For example, a well-known "dissection" technique in statistics is that of *diagnostics*, namely, taking a model apart to try to find out where its limitations are. In regression, for example, the prediction errors (the residuals) may be plotted or analyzed as function of a specific input variable X_i. If there is evidence of a systematic dependence (e.g., larger errors for larger values of X_i) then it may suggest how the model can be improved. Statisticians traditionally rely on a variety of such diagnostic techniques (including visualization) to provide insight and feedback in an iterative manner: model, fit, evaluate, model, fit, evaluate, and so forth. In the "massive data world" of data mining it is not always practical to apply such diagnostics, yet there is clearly substantial room for improvement in developing tools and techniques that allow a user to interact with both data and models in an iterative manner. In an ideal world this is certainly preferable to treating each data mining algorithm as a stand-alone black-box where the actions of the user are limited to tweaking various algorithmic parameters such as search options and so forth, rather than being allowed to explore the interaction of the data and model more directly.

6. Success Stories from the Interface

There can be significant differences between a typical computer scientist's view of data and a typical statistician's view of the same data. Despite this (or indeed perhaps because of this difference) there are numerous well-known examples of symbiosis at computer science/statistics interface. For example, there appears to be a common recurring pattern where research on essentially the same idea is carried out independently within statistics and within computer science, and both sets of ideas are then subsequently integrated to form a much richer and broader framework. We discuss briefly below a few well-known recent examples of this pattern in the general context of machine learning and data mining:

- Early work on neural networks focused largely on representational properties and biological plausibility of the models and details of training algorithms [RM86]. A broader statistical view of neural networks as highly flexible non-linear regression models gradually began to emerge, both from within the neural network community (e.g., [Bis95]) and from members of the statistical community who had taken an interest in this new form of regression model [GBD92, CT94, Rip94]. For example, links to more established statistical ideas such as generalized linear models and projection pursuit regression led to new models and algorithms that are "hybrids" of statistical and neural network research (e.g., [JJ94]).

- Graph-based models for efficient representation of multivariate distributions had been known for some time in areas such as genetics (e.g., [CTS78]). In the late 1980's more general and widely-applicable frameworks were developed independently within statistics (e.g., the acyclic directed graphical model framework of [LS88]) and the largely equivalent belief network framework pioneered by Pearl [Pea88] within computer science. The last 10 years has seen substantial research activity within both communities, particularly in the area of learning such models directly from data (e.g., [Hec95, Jor98]). The recent work on learning graphical models provides an excellent example of the symbiosis of computer science and statistics: efficient graph-based algorithms for computational inference (computer science) coupled with general principles for parameter and structure estimation from data (statistics).

- Latent (hidden) variable models have a long history in statistics (e.g., see [DT99]). Work on this topic within computer science is more recent, but has had a broad impact in the last 20 years in a variety of applications: hidden Markov models in speech recognition, latent variable models in language and text modeling, mixture-models and clustering techniques, and neural networks with hidden variables [HS99]). The EM procedure, an inherently computational concept but one that is motivated by some very fundamental principles in statistical inference, has played a pivotal role in all of these computer science and engineering applications.

- Decision trees were originally developed in applied statistics in the 1960's (e.g., [MS63]) but did not receive much attention within statistics until the publication of the pioneering work on CART [BFOS94]. Quinlan independently popularized the use of trees in machine learning with his ID3 and C4.5 family of algorithms [Qui87, Qui93]. Both the CART and ID3/C4.5 approaches share many common ideas, resulting in quite similar tree-learning algorithms. The statistical work on trees typically emphasizes parameter estimation and tree selection aspects of the problem, while more recent work on trees in data mining has emphasized data management issues (e.g., [GGRL99]). Decision trees are now a cornerstone in the toolbox of every data miner, with many relatively minor variants of tree algorithms, but all having much in common with the original ideas from the CHAID, CART, and ID3 research.

- Boosting algorithms are a class of techniques for iteratively constructing more complex predictive models by combining simpler models. Boosting was originally proposed within the framework of computational learning theory [FS97], a sub-field of machine learning concerned with theoretical analyses of learning algorithms. (Although having much in common, computational learning theory differs in many aspects from statistics, for example, emphasizing worst-case distribution-free learning models in contrast to the more typical average-case learning models in statistics). Subsequent empirical work revealed that boosting was not only of theoretical interest, but that it provided a very useful practical tool for improving the generalization performance of a wide variety of prediction models [BK99]. Statisticians began to take an interest in boosting, resulting both in powerful new algorithms [Fri99, Rid00] as well a general statistical theory that explains when and why the method works [FHT00].

The point of the examples above is to show that there is a long and successful tradition of "marrying" ideas, theories, and techniques developed relatively independently within computer science and within statistics. Naturally since computer science is a much younger discipline than statistics, the field of statistics has a much broader scope (in the context of learning from data). For example, there are large areas of data analysis such as spatio-temporal modeling, repeated measures/longitudinal data, and so forth, where data mining and machine learning have not had any appreciable impact. On the other hand, there are areas where a computational approach to learning has added concepts to data analysis that are relatively unrelated to anything in statistics. For example, Vapnik's theory of generalization based on margins [Vap98], and the subsequent development of support vector machines based on this theory [SBS99] could be viewed as being quite distinct from conventional statistical thinking on how to build predictive models. Despite these differences, the two fields nonetheless have much in common and data mining can prosper by cultivating and harvesting ideas at the interface.

7. The Dark Side of the Interface

> Data snooping occurs when a given set of data is used more than once for the purposes of inference or model selection. When such data reuse occurs, there is always the possibility that any satisfactory results may simply be due to chance rather than to any merit inherent in the method yielding the results.

This quote from [STW99] is an example of a very different view of data mining: a chapter on data mining and statistics would not be complete without reference to this viewpoint. Terms such as "data snooping," "data mining," "data dredging", and "data fishing" have historically been used in statistics to describe the use of computational search techniques to overfit data sets and uncover patterns of spurious origin (e.g., [Arm67, SS66, Lov83, Cha95b]). This interpretation of the term"data mining" is still widely held in areas such as econometrics (e.g., [Hen95, Whi01]). Clearly this is an entirely different view of data mining to the "computer science interpretation" we have been discussing so far in this chapter. Nonetheless this view has some interesting implications for the modern data miner that we discuss in more detail below.

7.1. Variable Selection for Regression

A classic example of data mining in this context occurs in variable selection for regression (e.g., [Mil90]), especially when applied to a relatively small training data set with no data used for holdout testing. Lets say we are trying to fit a simple linear regression model of the form

$$Y = \alpha_0 + \alpha_1 X_i + \alpha_2 X_j + e, \quad 1 \leq i, j \leq d, \ \ i \neq j, \tag{3.4}$$

where Y is the real-valued variable we are trying to predict, the α's are (unknown) parameters of the model, e is additive noise, and X_i and X_j are two particular variables from a set of d variables available to us. So the problem is to find the "best" pair of variables from the set of all possible pairs in the set of d variables. Here "best" is judged by minimum squared error (or some such similar score function) between the model's predictions and the observed Y's. Let ρ be the empirical linear correlation coefficient between the model's predictions and the actual observed Y values, i.e.,

$$\rho = \frac{\frac{1}{n}\sum_{l=1}^{n}(y_l - \hat{y}_l)^2}{\sigma_y \sigma_{\hat{y}}}$$

where y_l is the target value for the lth training data point and $\hat{y}_l$ is the model's prediction based on the lth set of input values, $1 \leq l \leq n$, and where σ_y and $\sigma_{\hat{y}}$ are the standard deviations of the target values and the predictions respectively. ρ close to 1 (or -1) indicates high positive (or negative) linear correlation, and ρ close to 0 indicates little or no correlation. We can rephrase our problem as that of finding X_i and X_j with the highest positive correlation, i.e., $(i, j) = \arg\max_{i,j}\{\rho_{ij}\}$, where ρ_{ij} is the correlation between y and $\hat{y}$ using the best-fitting linear predictor with variables X_i and X_j. From a computational viewpoint searching for the best pair is an interesting search problem and systematic search strategies such as branch-and-bound can be quite useful.

However, ignoring the details of how the search would be performed, consider the effect of the size of d, the total number of variables available. As d increases, the chances of selecting a pair of variables that *appear* to predict Y well, but in fact are not good predictors, also increases. To see why this can happen, consider for example the case where none of the possible pairs of X variables have any predictive power at all! i.e., all predictions are essentially random noise or equivalently $p(Y|X_i, X_j) = p(Y), \forall i, j$. Of course as data analysts we do not actually know (or even believe) that this is the case, since if we really believed there was no dependence we would not be trying to fit a predictive model. Nonetheless most data miners who have worked on real-world data sets are well aware that we occasionally encounter applications where the available predictor variables appear to have virtually no predictive power.

In the context of such a data set, a significant problem lies in the fact that ρ_{ij} will vary across all (i, j) pairs and will vary from data sample to data sample. Thus, the *maximum* of the empirically observed ρ_{ij}'s is also a random quantity with its own distribution, given a particular data set. Even if the true expected value across different data sets of each individual ρ_{ij} is zero (i.e., the model has no predictive power for each i and j), the expected value of the empirically observed *maximum* over the set of ρ_{ij}'s will be non-zero and positive (i.e., it will be optimistically biased). As the size of this set increases (as the number of candidate variables d is increased) the expected value of the maximum will also increase, i.e., we will tend to find better and better-looking regressions even though no real predictive power exists. This specific problem of spurious correlations has received particular attention in the fields of social science and econometrics [Ein72, Lea78, Hen95].

Other related (but more subtle) versions of "data-dredging" also exist. Imagine, for example, that we have multiple different data sets available to us, perhaps different derived sets of features, sets of features derived for different time-windows (for time-dependent variables), and so forth. Once again, imagine that none of these features have any predictive power. By sequentially fitting models and performing model selection on these different sets of features, we gradually increase our chances of accepting a model that happens to look like it has good predictive power, when in fact it has none.

7.2. Model Selection using Validation Data Sets

Now of course most current data miners are well aware of these sorts of problems. For example, the use of validation data sets and cross-validation techniques are widely used in data mining to provide more accurate estimates of a model's true generalization capabilities. Nonetheless, similar problems still lurk in the background, even with the use of independent training and validation data sets. For example, consider a simple binary classification problem, with N examples, d binary input variables (attributes) $X_1, \ldots, X_d$, and a single binary (target) class variable C. Say we train a variety of different well-known classification models $M_1, \ldots, M_m$ (e.g., decision trees, support vector machines, naive Bayes) with all kinds of variations (boosted versions, different feature subsets, combined models, etc)—modern data mining and statistical packages make it relatively easy to train many different models.

Naturally the models with more degrees of freedom will tend to fit the training data better, so classification error on the training data is not a good guide to how well the model will perform on new data. Since in many data mining applications we are fortunate to have large data sets it is easy to keep aside a separate validation data set for model selection. We estimate the classification error of each of our m models on the validation data, $e_1, \ldots, e_m$, and select the model with the lowest error rate $e_{\min} = \min\{e_1, \ldots, e_m\}$ as the model we will use for future predictions.

Now let us assume that there is no predictive power at all in any of the d input variables, so that one can do no better than random predictions with *any* classifier using these same variables as inputs, i.e., the error rate = 0.5 (assuming the two classes are equally likely). This "best achievable" error rate is known as the Bayes error rate for a given set of input variables. Note again that $e_{\min}$ is a random quantity in the sense that from one data set to the next it will vary. $e_{\min}$ for a particular pair of training/validation data sets and a particular set of m classifiers will typically be less than 0.5, and perhaps significantly less than 0.5 depending on how many models we try and how many data points are available. We can think of each of the m classifiers as simply adding in a column of predicted Y values for the validation data set. These columns are not independent since they all are based on models built on the same training data, but to a first approximation can be thought of as being independent if the true underlying relationship between the inputs X and the class variable C is random. We can imagine sorting the observed class values in the validation data set into two subsets of rows, subset A where C takes value 0 and subset B where C takes value 1. Keeping in mind that each of our trained classifiers will be no better than black boxes that randomly produce 0's and 1's (with about equal probability), the validation error rate for each model is simply the number of 1's produced in subset A and the number of 0's produced in subset B, divided by the total number of rows N_v. Essentially we can view this as a version of a binomial random process, no different that tossing a coin N_v times and counting the occurrence of 0's and 1's in certain locations. As N_v goes to infinity, we know by the law of large numbers that indeed the empirically-observed error rate for each column will get closer to the true value of 0.5. But keep in mind that (a) we only have a finite amount of data N_v for validation, and (b) that we are in fact selecting the column with the *minimum* empirical error rate. Clearly, this minimum error rate will tend to be less than 0.5 (i.e., it is optimistically biased) and the expected value of this negative bias (if we repeated this experiment many times) will be a function of both N_v and m, the number of models being used. In extreme cases (small N_v and large m) it could be very biased, e.g., we might find a model with validation error of only 0.3 when the true error rate of all models is 0.5.

Now of course in practice the inputs will usually have *some* predictive power. However, the validation estimate will still be optimistically biased, precisely because it is defined as the minimum over a set of random quantities. The point here is that while validation data and holdout data provide useful "insurance policies" that usually provide much better guidance than performance on training data alone, the element of chance and variability is nonetheless still always present. A useful exercise is to always imagine what your data mining algorithm might do if it were presented with truly random data (as in the examples above). For example, imagine what might happen if a standard association rule algorithm were run on data where all the items (columns) were generated in an independent manner, i.e., there are truly no rules relating any of the items. The exact number of rules found and their strengths will depend on how many items p and how many transactions n there are, but the general effect is the same: even on purely random data, the effect of massive search will be to find spurious associations. Note that although we might use hypothesis testing techniques to guard against such "noise," repeated application of multiple hypothesis tests on a massive scale will eventually lead to acceptance of false hypotheses. It is the massive search (the massive systematic search through itemsets) that is the root cause of this fundamental inability to distinguish pure noise from real structure in the data.

In practice we know that real structure often does exist (i.e., we are not dealing with random data) and that this will tend to dominate the random noise. Nonetheless the general point is that the use of unfettered search techniques can in theory get a data miner into trouble. It is the wise data miner that is aware of these risks and conveys a similar note of caution to the *user* of his/her algorithms, such as the unsuspecting business person or scientist who uses a data mining algorithm in "black box" fashion. Data miners need to be aware of the fundamental statistical aspects of inference from data (see [Jen91, Sal97, JC00] for further discussion). Data mining algorithms should not be a substitute for statistical common sense.

8. Conclusions

This chapter has argued that statistical and algorithmic issues are both important in the context of data mining. We have seen that there is a substantial history of successful research at the interface of computer science and statistics, despite the fact that computer scientists and statisticians have substantially different "cultural biases" in terms of how they think about data. For data miners the message is clear: statistics is an essential and valuable component for any data mining exercise. The future success of data mining will depend critically on our ability to integrate techniques for modeling and inference from statistics into the mainstream of data mining practice.

Acknowledgements

The author gratefully acknowledges valuable feedback on earlier drafts of this chapter from Tom Dietterich and Greg Ridgeway as well as from two anonymous reviewers. This work was supported in part by the National Science Foundation under Grant IRI-9703120.

Notes

1. Note that "score" is used in statistics in a much more specific sense, namely, the derivative of the log-likelihood: here we purposely use "score" in a much broader sense. Readers familiar with this usage might wish to replace "score function" with "loss function" throughout

References

[AIS93] Agrawal, R., Imielinski, T., and Swami, A. (1993) Mining associations between sets of items in massive databases, in *Proceedings of the 1993 ACM SIGMOD International Conference on the Management of Data*, New York, NY: ACM Press, 207–216.

[Arm67] Armstrong, J. S., (1967) Derivation of theory by means of factor analysis or Tom Swift and his electric factor analysis machine, *American Statistician*, 21, 415–22.

[BK99] Bauer, E. and Kohavi, R. (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning*, 36(1/2), 105–139.

[BP99] Bay, S. and Pazzani, M. (1999) Detecting change in categorical data: mining contrast sets, in *Proceedings of the Fifth ACM International Conference on Knowledge Discovery and Data Mining*, New York, NY: ACM Press, 302–305.

[Ben99] Bengio, Y. (1999) Markovian models for sequential data, *Neural Computing Surveys*, 2, 129–162.

[BL00] Berry, M. J. A. and Linoff, G. (2000) *Mastering Data Mining: The Art and Science of Customer Relationship Management*, New York, NY: John Wiley and Sons.

[BD77] Bickel, P. J. and Doksum, K. A. (1977) *Mathematical Statistics: Basic Ideas And Selected Topics*, San Francisco, Holden-Day.

[Bis95] Bishop, C. (1995) *Neural Networks for Pattern Recognition*, Oxford, UK: Clarendon Press.

[BPR98] Bradley, P., Fayyad, U. M., and Reina, C. (1998) Scaling EM (expectation-maximization) to large databases, Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA.

[BFOS94] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., (1984) *Classification and Regression Trees*, Belmont, CA: Wadsworth Statistical Press.

[CGS00] Cadez, I., Gaffney, S., and Smyth, P. (2000) A general probabilistic framework for clustering individuals, in *Proceedings of the ACM Seventh International Conference on Knowledge Discovery and Data Mining*, New York, NY: ACM Press, 140–149.

[CTS78] Cannings, C., Thompson, E. A., and Skolnick, M. H. (1978) Probability functions on complex pedigrees, *Advances in Applied Probability*, 10, 26–61.

[CB90] Casella, G. and Berger, R. L. (1990) *Statistical Inference*, Wadsworth and Brooks.

[Cha95] Chatfield, C. (1995) *Problem Solving*, 2nd ed., Chapman and Hall.

[Cha95b] Chatfield, C. (1995) Chatfield, C. (1995) Model uncertainty, data mining and statistical inference, *Journal of the Royal Statistical Society A*, pp. 415–466.

[CT94] Cheng, B. and Titterington, D. M. (1994) Neural networks: a review from a statistical perspective, *Statistical Science*, 9, 2–54.

[CS81] Cox, D. R. and Snell, E. J. (1981) *Applied Statistics: Principles and Examples*, London: Chapman and Hall.

[DH00] Domingos, P. and G. Hulten (2000) Mining high-speed data streams, in *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pp. 71-80, Boston, MA: ACM Press.

[DVJ99] Du Mouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. (1999) Squashing flat files flatter, in *Proceedings of the Fifth ACM International Conference on Knowledge Discovery and Data Mining*, New York, NY: ACM Press, 6–15.

[DT99] Dunmur, A. P. and Titterington, D. M. (1999) Analysis of latent structure models with multidimensional latent variables, in *Statistics and Neural Networks : Advances at the Interface*, J.W. Kay and D.M. Titterington (eds.), New York: Oxford University Press, 165–194.

[Ein72] Einhorn, H. (1972) Alchemy in the behavioral sciences, *Public Opinion Quarterly*, 36, 367–378.

[EP96] Elder, J. F., and Pregibon, D. (1996) A statistical perspective on knowledge discovery in databases, in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, eds., Cambridge, MA: The MIT Press, pp. 83–115.

[ED96] Everitt B. and Dunn G. (1996) *Applied Multivariate Data Analysis*, New York, NY: John Wiley and Sons.

[FS97] Freund, Y. and Schapire, R. (1997) A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, 55(1), 119–139.

[Fri99] Friedman, J. H. (1999) Greedy function approximation: a gradient boosting machine, Technical Report, Statistics Department, Stanford University.

[FF99] Friedman, J. H. and Fisher, N. I. (1999) Bump hunting in high-dimensional data, *Statistics and Computing*, 9, 123–143.

[FHT00] Friedman, J. H., Hastie, T., and Tibshirani, R. (2000), Additive logistic regression: a statistical view of boosting, *Annals of Statistics*, 28(2), 337–374.

[GGRL99] Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W-Y. (1999) BOAT—Optimistic decision tree construction. *Proceedings of the SIGMOD Conference 1999*, New York, NY: ACM Press, 169–180.

[GBD92] Geman, S., Bienenstock, E., and Doursat, R. (1992) Neural networks and the bias/variance dilemma, *Neural Computation*, 4, 1–58.

[GMPS96] Glymour C., Madigan D., Pregibon D., Smyth P. (1996) Statistical inference and data mining, *Communications of the ACM*, 39(11), 35–41.

[GMPS97] Glymour C., Madigan D., Pregibon D., Smyth P. (1997) Statistical themes and lessons for data mining, *Journal of Data Mining and Knowledge Discovery*, 1, 11–28.

[GEC84] Goodhardt, G. J., Ehrenberg, A. S. C., and Chatfield, C. (1984) The Dirichlet: a comprehensive model of buying behavior, *J. R. Statist. Soc. A*, 147(5), 621–655.

[Han94] Hand, D. J. (1994) Deconstructing statistical questions, *J. R. Statist. Soc. A*, 157(3), 317–356.

[Han98] Hand D. J. (1998) Data mining—statistics and more, *The American Statistician*, 52(2), 112–118.

[HMS01] Hand, D. J., Mannila, H., and Smyth, P. (2001) *Principles of Data Mining*, Cambridge, MA: The MIT Press, forthcoming.

[Hec95] Heckerman, D. (1995) A tutorial on learning Bayesian networks, Techical Report MSR-TR-95-06, Microsoft Research, Redmond, WA.

[HCM00] Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadie, C. (2000) Dependency networks for density estimation, collaborative filtering, and data visualization, Technical Report MSR-TR-2000-16, Microsoft Research, Redmond, WA.

[Hen95] Hendry, D. F. (1995) *Dynamic Econometrics*, New York, NY: Oxford University Press.

[HS99] Hinton, G.E., and T. Sejnowski (eds.) (1999) *Unsupervised Learning: Foundations of Neural Computation*, The MIT Press.

[Hof99] Hoffmann, T. (1999) Probabilistic latent sematic indexing, *Proceedings of SIGIR '99*, 50–57.

[HC78] Hogg, R. V. and Craig, A. T. (1978) *Introduction to Mathematical Statistics*, 4th ed. Macmillan.

[HPS97] Hosking, J. R. M., Pednault, E. P. D., and Sudan, M. (1997) A statistical perspective on data mining, *Future Generation Computer Systems*, 13, 117-134.

[Jen91] Jensen, D. (1991) Knowledge discovery through induction with randomization testing, in *Proceedings of the 1991 Knowledge Discovery in Databases Workshop*, G. Piatetsky-Shapiro (ed.), Menlo Park, CA: AAAI Press, 148–159.

[JC00] Jensen, D. and Cohen, P. (2000) Multiple comparisons in induction algorithms, *Machine Learning*, 38, 309–338.

[Joh99] John, G. (1999) personal communication.

[JJ94] Jordan, M. I. and Jacobs, R. A. (1994) Hierarchical mixtures of experts and the EM algorithm, *Neural Computation*, 6, 181–214.

[Jor98] Jordan, M. I. (ed.) (1998) *Learning in Graphical Models*, Cambridge, MA: The MIT Press.

[Kle98] Kleinberg, J. M. (1998) Authoritative sources in a hyperlinked environment, in *Proc. of ACM-SIAM Symp. on Discrete Algorithms*, 668–677.

[Kni00] Knight, K. (2000) *Mathematical Statistics*, Chapman and Hall.

[Koh00] Kohavi, R. (2000) personal communication.

[Lam00] Lambert, D. (2000) What use is statistics for massive data?, preprint.

[LS88] Lauritzen, S. L. and Spiegelhalter, D. J. (1988) Local computations with probabilisties on graphical structures and their application to expert systems (with discussion), *J. Roy. Statist. Soc. B*, 50, 157–224.

[Lea78] Leamer, E. E. (1978) *Specification Searches: Ad Hoc Inference with Non-Experimental Data*, New York, NY: John Wiley.

[LB97] Letsche, T. A. and Berry, M. W. (1997) Large-scale information retrieval with latent semantic indexing, *Information Sciences—Applications*, 100, 105–137.

[Lov83] Lovell, M. (1983) Data mining, *Review of Economics and Statistics*, 65, 1–12.

[MTI95] Mannila, H., Toivonen, H. and Inkeri Verkamo, A. (1995) Discovering frequent episodes in sequences, in *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA: AAAI Press, 210–215.

[Mil90] Miller, A. J. (1990) *Subset Selection in Regression*, London, Chapman and Hall.

[MF01] Moe, W. W., Fader, P. (2001) Which visits lead to purchases? dynamic conversion behavior at e-commerce sites, Working Paper 00-023, Department of Marketing, Wharton School of Business, University of Pennsylvania.

[Moo99] Moore, A. W., (1999) Cached sufficient statistics for automated discovery and data mining from massive data sources, online white paper, Department of Computer Science, Carnegie Mellon University.

[ML98] Moore, A. W. and Lee, M. (1998) Cached sufficient statistics for efficient machine learning with large data sets, *Journal of Artificial Intelligence Research*, 8, 67–91.

[MS63] Morgan, J. N. and Sonquist, J. A. (1963) Problems in the analysis of survey data and a proposal, *J. Am. Stat. Assoc*, 58, 415–434.

[Pea88] Pearl, J. (1988) Probabilistic Inference in Intelligent Systems: Networks of Plausible Inference, San Mateo, CA: Morgan Kaufmann.

[PK99] Provost, F. and Kolluri, V. (1999) A survey of methods for scaling up inductive algorithms, *Journal of Data Mining and Knowledge Discovery*, 3(2), 131–169.

[Qui87] Quinlan, J. R. (1987) Generating production rules from decision trees, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 304–307.

[Qui93] Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*, San Mateo: CA, Morgan Kaufmann.

[Rid00] Ridgeway, G., (2000) Prediction in the era of massive data sets, *Statistical Modeling for Data Mining*, P. Giudici (ed.), Kluwer, 109–119.

[Rip94] Ripley, B. D. (1994) Neural networks and related methods for classification (with discussion), *J. R. Statist. Soc. B*, 56, 409–456.

[Rip96] Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*, Cambridge, UK: Cambridge University Press.

[Ros00] Ross, S. M. (2000) *Introduction to Probability Models*, San Diego, CA: Academic Press.

[RM86] Rumelhart, D. E. and McClelland, J. L. (eds.) (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, Cambridge, MA: The MIT Press.

[Sal97] Salzberg, S. L. (1997) On comparing classifiers: pitfalls to avoid and a recommended approach, *Data Mining and Knowledge Discovery*, 1:3, 317–327.

[SBS99] Scholkopf, C., Burges, J. C., and Smola, A. J. (1999) *Advances in Kernel Methods*, Cambridge, MA: MIT Press.

[SS66] Selvin, H. and Stuart, A. (1966) Data dredging procedures in survey analysis, *American Statistician*, 20(3), 20–23.

[SG92] Smyth, P. and Goodman, R. (1992) An information-theoretic approach to rule induction from databases, *IEEE Transactions on Knowledge and Data Engineering*, 4(4), 301–306.

[Smy00] Smyth, P. (2000) Data mining: data analysis on a grand scale?, *Statistical Methods in Medical Research*, 9, 309–327.

[STW99] Sullivan, R., Timmermann, A. and White, H. (1999) Data snooping, technical trading rule performance, and the bootstrap, *Journal of Finance*, 54, 1647–1692.

[Vap98] Vapnik, V. (1998) *Statistical Learning Theory*, New York, NY: Springer Verlag.

[WB86] Walker, M. G. and Blum, R. L. (1986) Towards automated discovery from clinical databases: the RADIX project, in *Proceedings of the Fifth Conference on Medical Informatics*, volume 5, 32–36.

[WMB99] Witten I. H., Moffat A., Bell T. C. (1999) *Managing gigabytes: compressing and indexing documents and images.* San Francisco, CA: Morgan Kaufmann, (2nd ed.).

[WK98] Wedel, M. and Kamakura, W. A. (1998) *Market Segmentation: Conceptual and Methodological Foundations*, Boston, MA: Kluwer Academic Publishers.

[Whi01] White, H. (2001), A reality check for data snooping, *Econometrica*, forthcoming.

[ZRL97] Zhang, T., Ramakrishnan, R., Livny, M., (1997) BIRCH: A new data clustering algorithm and its applications, *Journal of Data Mining and Knowledge Discovery*, 1(2), 141–182.

Chapter 4

MINING LARGE IMAGE COLLECTIONS

Michael C. Burl

Abstract NASA has been involved with remote exploration of the solar system for over forty years and, as a result, has accumulated a vast archive of images. Continued improvements in acquisition and storage technology are yielding new image sets with data volumes measured in terabytes. Within these large image collections there is a wealth of scientific information, but getting from the data to knowledge is a difficult problem both due to the size of the datasets involved and the difficulty of automatically interpreting image data. This chapter provides an overview of our efforts to develop algorithms for mining useful information from large image collections.

Keywords: Diamond Eye, image mining, recognition, discovery, query-by-content.

1. Introduction

Image mining can be defined loosely as the use of automated or semi-automated systems to transform large sets of raw images into semantically-meaningful knowledge. Improvements in acquisition and storage technology (inexpensive digital cameras and scanners, large disk drives, CD-writers, etc.) have led to an explosion in the number and size of image collections in a variety of fields, making automated approaches to knowledge extraction from images critically important. Nowhere is this more true than at NASA, where large image collections are often *the* primary product of spacecraft missions. The final goal, however, is not simply to collect and archive images, but to use the images to support scientific investigations into a wide variety of processes and phenomena. In this respect, more data at higher resolution (spatial, temporal, and spectral) is always desirable from the scientist's perspective; however, the data volume has already surpassed the point where comprehensive manual analysis of a complete image collection is feasible.

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 63–84.
© 2001 *Kluwer Academic Publishers.*

As one example, the Magellan mission to Venus [PFJ+91] in the early 1990's returned more data than all previous planetary missions combined [SSA+92] covering 98% of the surface of Venus at resolutions up to 120m × 120m (with some areas imaged multiple times). Based on earlier lower-resolution imagery, scientists expected to find approximately one million small volcanoes in the Magellan dataset [AS90]. A comprehensive catalog including the size, location, and other relevant information (e.g., sub-type) for each volcano is clearly essential for supporting advanced studies such as global cluster analysis of volcano locations and examining the relationship of volcanism with local tectonic structure. However, it was estimated that the process of manually transforming the raw image collection into such a catalog would require twenty man-years. The JARtool project (JPL Adaptive Recognition Tool) [BAS+98, BFP+94] was initiated as a pioneering effort to develop and apply image mining ideas to this problem. Lessons learned from JARtool were later leveraged into the design of a second generation image mining architecture, known as Diamond Eye.

Diamond Eye [BFR+99b] is a Java-based system that enables users (scientists) to locate and catalog objects of interest in image collections. The system is based on a distributed applet/server architecture that provides platform-independent access to image mining services. Each server program is typically co-located with a large image repository to enable mining the data in place. The servers are also coupled with an object-oriented database and a computational engine such as a network of high-performance workstations (NOW). The database provides persistent storage and enables querying of the "mined" information, while the computational engine provides parallel execution of the most demanding parts of the data-mining task: image processing, object recognition, and query-by-content operations. Through a middleware interface, the core mining algorithms, which are written in "fast languages" (e.g., C or C++), are tied into the system. Diamond Eye provides an application-independent infrastructure into which domain-specific mining algorithms can be easily inserted.

The fundamental challenge in developing image mining algorithms is determining what mathematical processing should be applied to the low-level pixel representation contained in a raw image (or image sequence) in order to identify spatial objects, such as volcanoes and craters, or dynamic events, such as eruptions or satellite motion. As a coarse taxonomy, we view algorithms as lying on a spectrum indexed by the amount of a priori knowledge (number of training examples) used or required. At one extreme, *recognition algorithms* use a large number of examples to derive precise statistical models and then use these models to locate other instances of the target object. At the other extreme, *discovery algorithms* use essentially no examples, but instead rely on generic assumptions to look for "interesting" objects without having a specific target object in mind. Between the two extremes, *query algorithms* use one or a few examples or even user sketches to formulate a coarse model that can be used to conduct an ad hoc search for some instances of an object of interest. Selected Diamond Eye algorithms from each of these three categories will be described in detail.

2. Recognition Algorithms

The algorithm used in the JARtool system provides a good illustration of a statistically-driven recognition algorithm. Rather than being hand-coded specifically for volcano detection, the algorithm uses scientist-provided examples to learn a volcano appearance model. The algorithm consists of three components: focus of attention (FOA), feature extraction, and classification. During a learning phase, the algorithm is provided with labeled training images that it uses to derive the parameters needed by each of the components. During the production phase, these parameters are "locked in" and the resulting models are applied to new images to identify novel instances of the target objects.

2.1. Focus of Attention (FOA)

The first component in the algorithm is a focus of attention (FOA) mechanism that is designed to take as input an image and produce as output a discrete list of *candidate* object locations. In principle, every pixel in the image could be treated as a candidate location; however, this is too expensive computationally. Instead, the FOA is used to quickly exclude areas that clearly have no target objects. Only local regions passing the FOA are examined by subsequent processes; hence, the FOA must operate in a low-miss-rate regime. The rate of false positives at this stage is not critical, since these may be rejected later by the classification stage.

Empirically we have found that a matched filter, i.e., a linear filter that matches the signal one is trying to find, provides a reasonable FOA method. Of course, this approach is too simplistic for generic recognition problems but works well in the volcano domain because of special characteristics of the problem. The matched filter is defined as follows. Let $\mathbf{v}_i$ denote a $k \times k$ pixel region around the i-th training example. For volcanoes, there is some loss of information due to this windowing process; however, since much of the useful information is concentrated at the center of the volcano (for example, the transition in shading and presence of a summit pit), the approach is still effective. Each $k \times k$ region can be normalized with respect to the local average image brightness (DC level) and contrast as follows:

$$\tilde{\mathbf{v}}_i = \frac{\mathbf{v}_i - \mu_i \cdot \mathbf{1}}{\sigma_i} \tag{4.1}$$

where μ_i is the mean of the pixels in $\mathbf{v}_i$, σ_i is their standard deviation, and $\mathbf{1}$ is a $k \times k$ matrix of ones. In most cases, this normalization is essential since there may be large fluctuations in the DC and contrast between images and even between local areas of the same image. A modified matched filter $\mathbf{f}$ is constructed by averaging the normalized examples from the training set.

Applying the matched filter to an image involves computing the normalized cross-correlation of $\mathbf{f}$ with each $k \times k$ image patch. High response values indicate that there is strong correlation between the filter and the image patch. Candidate locations are determined by thresholding the response values and spatially aggregating any threshold crossings that are within a prescribed distance from each other. More sophisticated FOA models using clustering and multiple matched filters are also possible [SB97], but did not provide significant improvement in the volcano application.

2.2. Feature Extraction

A region of interest (ROI) identified by the focus of attention algorithm can be viewed as a point in a k^2-dimensional space by stringing the $k \times k$ pixel values out into a long vector. Note, however, that there is a loss of spatial neighborhood information. Algorithms that treat the data in this form will not *explicitly* know that certain pixels were adjacent in the original image data. Also, given a relatively small number of training examples relative to the dimensionality of the data, there is little hope of learning anything useful without additional constraints. In the volcano application, experimental results with a variety of feedforward neural network classification models verified this hypothesis [Bal94]; the training data (several hundred of examples in 225 dimensional space) were often linearly separable in pixel space resulting in an underconstrained training procedure that allowed the neural network models to essentially memorize the training data perfectly, but with poor generalization to unseen data.

To work around a relatively small number of training examples, we make use of the fact that for visual data, there is additional prior information that helps constrain the problem. Specifically, there is reason to believe that visually coherent objects "live" on low-dimensional manifolds embedded in k^2-dimensional space. Although the manifold of a particular object is certainly nonlinear, the principal components analysis (PCA) paradigm can be used to approximate the manifold with a low-dimensional hyperplane. This approximation can be viewed as a mapping from the high-dimensional pixel space to a lower dimensional feature space in which the features consist of linear combinations of the pixel values[1].

PCA can be summarized as follows. The goal is to find a q-dimensional subspace such that the projected data is closest in L_2 norm (mean square error) to the original data. This subspace is spanned by the eigenvectors of the data covariance matrix having the highest corresponding eigenvalues. Although a full covariance matrix often cannot be reliably estimated from the number of examples available, the approximate highest eigenvalue basis vectors can be be computed using singular value decomposition (SVD).

Each normalized training example is reshaped into a vector and placed as a column in an $n \times m$ matrix X, where n is the number of pixels in an ROI ($n = k^2$) and m is the number of examples. The SVD produces a factorization of X as follows:

$$X = USV^T \tag{4.2}$$

For notational convenience, we will assume m is less than n. Then in Equation 4.2, U is an $n \times m$ matrix such that $U^T U = I_{m \times m}$, S is $m \times m$ and diagonal with the elements on the diagonal (the singular values) in descending order, and V is $m \times m$ with $V^T V = VV^T = I_{m \times m}$. Notice that any column of X (equivalently, any ROI) can be written exactly as a linear combination of the columns of U. Furthermore, if the singular values decay quickly enough, then the columns of X can be closely approximated using linear combinations of only the first few columns of U. That is, the first few columns of U serve as an approximate basis for the entire set of examples in X.

Thus, the best (in the L_2 sense) q-dimensional subspace on which to project is spanned by the first q columns of U. An ROI is projected into this q-dimensional feature space as follows:

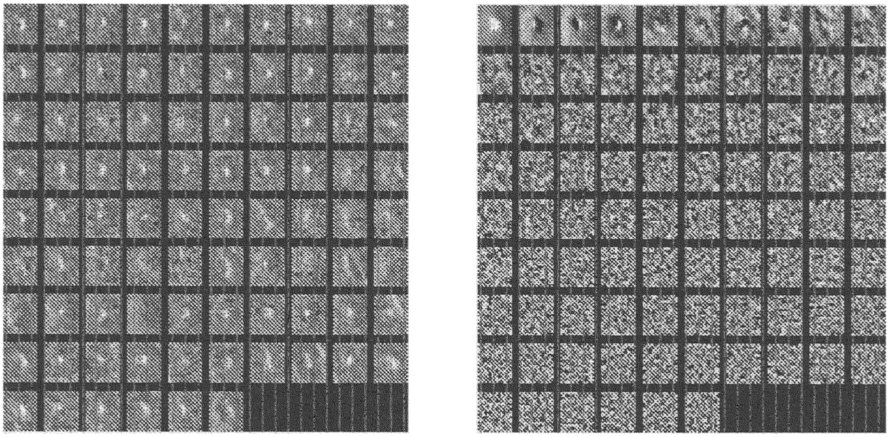
$$\mathbf{y} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \ldots & \mathbf{u}_q \end{bmatrix}^T \mathbf{x} \tag{4.3}$$

where $\mathbf{x}$ is the intensity-normalized ROI reshaped as an n-dimensional vector of pixels, $\mathbf{u}_i$ is the i-th column of U, and $\mathbf{y}$ is the q-dimensional vector of measured features.

Figure 1a shows a set of training volcanoes. Figure 1b shows the principal components, i.e., the columns of U, reshaped as ROIs. The templates are ordered according to singular value so that the upper left template corresponds to the maximum singular value. Notice that the first ten templates (top row) exhibit structure while the remainder appear very random. This suggests using a subspace of dimension ≤ 10. The singular value decay shown in Figure 1-c also indicates that 6 to 10 features will be adequate to encode most of the information in the examples. Indeed, parameter sensitivity experiments, which are reported in [BFPS96] show that values of q in the range 4–15 yield similar overall performance.

2.3. Classification

The FOA and feature extraction steps transform raw images into a discrete set of feature vectors. The final step is to classify the feature vectors as volcano or



(a) Training Volcanoes

(b) Principal Components

(c) Singular Values

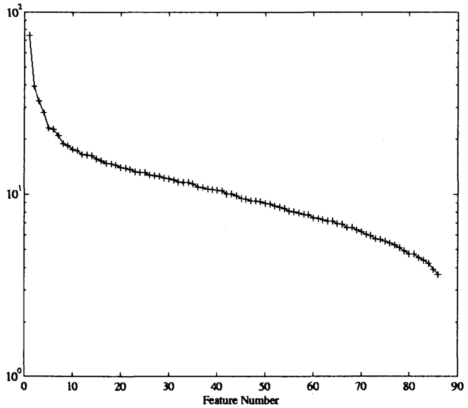


Figure 1. (a) The collection of volcanoes used for feature synthesis. (b) The principal components derived from the examples. (c) The singular values indicate the importance of each of the features for representing the examples.

non-volcano. Although FOA and feature learning are based exclusively on positive examples, the classifier uses both positive and negative training examples. In particular, the negative examples are regions of the training images identified by the FOA as promising, but which don't correspond to actual instances of the target object.

For the volcano problem, we experimented with a variety of algorithms including quadratic (or Gaussian) classifiers, decision trees, linear discriminant analysis, nearest neighbors using Euclidean and spatially weighted distance measures [Tur96], tangent distance [SYD93], kernel density estimation, Gaussian mixture models, and feedforward neural networks [AM97, Che96]. All of these methods (with the exception of linear discriminant analysis) yielded similar performance on an initial test set of images. The quadratic classifier was selected as the baseline since it is optimal for Gaussian data and provides posterior probability estimates, which can be thresholded to vary the trade-off between detection and false alarm rate. Letting ω_1 designate the positive class and ω_2 the negative class, Bayes' rule states:

$$p(\omega_i|\mathbf{y}) = \frac{p(\mathbf{y}|\omega_i) \cdot p(\omega_i)}{p(\mathbf{y}|\omega_1) \cdot p(\omega_1) + p(\mathbf{y}|\omega_2) \cdot p(\omega_2)} \tag{4.4}$$

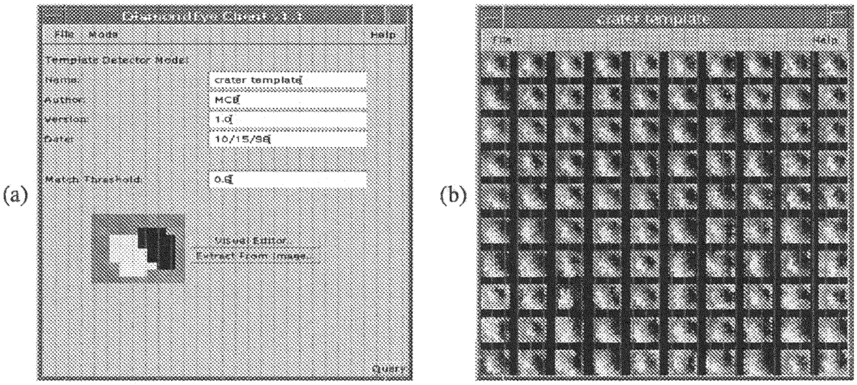
where $\mathbf{y}$ is the observed feature vector. For the quadratic classifier, the posterior probabilities are estimated by assuming the class-conditional densities are Gaussian

$$p(\mathbf{y}|\omega_i) = \mathcal{N}(\mathbf{y};\ \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{4.5}$$

where the statistics of each class ($\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$) are estimated from the labeled training data. Complete performance evaluations are given in [BAS$^+$98].

3. Query Algorithms

The use of learning techniques to construct precise recognizers depends on a relatively abundant supply of training examples. In many circumstances, however, it is desirable to search for objects based on a single example or even a notional concept of how an object should look. From a machine learning perspective, the task of generalizing from a single example in a multi-dimensional space seems ill-posed. However, there are two tricks we can employ to approach the problem. First, a priori knowledge about how an object might look under other circumstances can be used to create artificial training examples from the true example [DB00]. For example, knowledge that an object can be rotated and scaled a bit without changing the object's identity provides a means to synthesize new examples. Using a priori knowledge about invariances is closely related to the idea of "hints" in learning [AM95]. A second trick is to exploit the user as an oracle and use his knowledge to obtain relevance feedback (e.g., [SB90]). For example, suppose a coarse model derived from a single



Template Detector Model

Name: crater template

Author: MCB

Version: 1.0

Date: 10/15/98

Match Threshold: 0.6

Figure 2. (a) A QUERYBUILDER mode enables the user to construct visual query models of various types. The model shown is a template detector that can be used to find craters that look similar to the template. (b) Results of the query can be examined with the THUMBNAIL-BROWSER mode. For each thumbnail in the mosaic, there exists a ***linkback*** which makes it possible to go back to the specific image and region within the image from which the thumbnail was derived.

example can be used to pull other examples from the dataset along with a potentially large number of false examples. Feedback from the user as to which returned examples are valid can then be used to refine the model. Empirically, this bootstrapping process often converges to a reasonable model for the object of interest.

3.1. Continuously-scalable template models

Figure 2a illustrates one of the simpler forms of queries. The user provides a template that is similar in appearance to the desired target object either by extracting a region from an image (an existing example) or by hand-painting on a canvas with the mouse and a palette of gray levels. It is then possible to use this template and a normalized cross-correlation metric to assess the similarity between regions of an image and the template. Locations where the correlation is high are selected as potential matches. The best matches over a subset of Mars Viking images are presented to the user as thumbnails as in Figure 2b.

One idea for improving upon rigid template detectors involves using the steerable-scalable-deformable filtering ideas developed by Freeman and Adelson [FA91], Perona [Per95], Murase and Nayar [NBM96] and others. The basic idea is that a filter with parametric dependence [NBM96] can be approximately implemented using a finite set of basis filters. We have applied this approach to generate a scalable version of matched filtering that is useful when the primary variation in the appearance of an object is due to scale. This situation occurs if the distance of the imager from the object varies or if the object naturally occurs at different sizes.

The continuously-scalable template models (CSTMs) enable matching over a continuum of scales rather than just in a small range around the scale of the query example. The user-provided example is synthetically expanded into a "family" of templates by resampling the example at a number of different pixel densities. Conceptually, each of the family member should be used as a template for finding matches at a particular scale; however, the brute-force approach of correlating each family member against an image would be too expensive computationally. Instead, the template family is compressed using singular value decomposition (SVD). This process provides a reduced-dimensionality linear basis that approximately represents each family member. The basis can be used to efficiently calculate the correlation of an image with any family member and, therefore, provides a way to simultaneously detect and size instances of the object over a continuous range of scales. Coupling this technique with a pyramid representation of the images provides continuous coverage over orders of magnitude in scale. A generalization of this basic idea (including scaling, rotation, and affine "jitters") is described in [DB00].

3.2. Spatially deformable configuration of parts models

A more sophisticated form of query is based on the concept of a spatially deformable configuration of parts (SDCP; also called constellation model or factored appearance model). Originally developed as a recognition algorithm to be trained from examples, the key idea is to represent an object as a set of individual *parts* that are linked together through a probability distributions over their spatial arrangement [LBP95, BLP95, BLP96, BWP98, LBP98]. The probability distributions can be formulated in a special *shape space* that is invariant to geometrical deformations such as translation, rotation, scaling, and even affine transformations [LBP98]. An object is detected by applying part detectors that identify candidate locations for each part of the object. Candidate locations are then grouped into object hypotheses, which are evaluated based on their geometric arrangement. The SDCP model provides a powerful hybrid between purely image-based techniques (e.g., PCA, CSTM) and purely geometry-based techniques (e.g., geometric hashing [Wol90, RH95], invariants [MZ92], etc.)

To define an SDCP query model from a single example, the user marks regions of the object that should be used as parts. Here, the user employs some knowledge of how the part detectors work to insure that the parts selected will be distinctive and easy to locate in other instances of the object. The layout of the parts on the initial example is used as a coarse estimate of the object geometry. The system then hypothesizes an isotropic Gaussian noise model on the positions of the parts and uses this model to synthesize new geometrical arrangements. The new arrangements are presented to the user as cartoons. Relevance feedback from the user indicates which variations are plausible (still considered to be the same object) and which are not. The system incorporates the user feedback into a revised covariance estimate for the allowed deformations of the basic geometry. Empirically, a small number of iterations with 40 synthetic examples per iteration is sufficient for the system to converge to a reasonably accurate model of the geometric variation. The SDCP model can then be applied to a set of images to identify instances of the target object.

3.3. Relation to content-based image retrieval

We note that while there is a large content-based image retrieval (CBIR) thrust within the computer vision community [FSN$^+$95, PP96, Vir], the approach we use differs significantly. In the CBIR community, the focus is on retrieving images that are "visually similar". There is also an emphasis on making queries interactive so that answers are returned quickly (within a few seconds). Hence, many of the techniques are based on relatively low-level definitions of content, e.g., precomputed global properties of images such as color histograms and texture descriptors. In contrast, the Diamond Eye system is aimed at providing a "high-end" capability that can provide access to *specific objects* within images. In [FMW97] this distinction is described as finding "things" rather than "stuff". Due to the difference in emphasis, the Diamond Eye system employs more complex object-specific query models, but must rely on the availability of fast, parallel hardware to carry out its work.

4. Discovery Algorithms

Recognition and query algorithms are useful for finding particular objects. In some cases, however, we do not know what to expect or look for in advance, but would like to be able to identify any unusual or "interesting" objects that are present. This scenario is especially relevant in space exploration where spacecraft venture to new or poorly understood environments , e.g., future missions to Pluto, the subsurface of Europa, or the Kuiper Belt. Scientists analyzing Voyager fly-by imagery of Neptune's moon Triton discovered a set of geologic features, now believed to be "ice geysers" or "cryogenic volcanoes", that had not been seen elsewhere in the Solar System. As a second example, scientists recently unexpectedly discovered evidence for recent liquid water activity on Mars from the high resolution images taken with the Mars Observer Camera onboard the Global Surveyor spacecraft [ME00]. Also, any situation where the precise nature and appearance of the target object is unknown, but the target object can be presumed to be visually different than the surrounding background, is potentially amenable to discovery-type algorithms. Closer to home there are applications involving the identification of defects in textiles [CFA91, BB98, SSG99], tumors in medical imaging, vortices in fluid flow, etc. In most of these cases, we have seen examples of these objects before (e.g., we have all seen defects in textiles); however, there may not be enough statistical regularity in the appearance of the object to support an object-specific recognition model.

The use of the word "interesting" to describe the type of objects a discovery algorithm should find is a bit loaded since two observers with specific questions and hypotheses in mind may find that completely different portions of an image are interesting. We do not deal this type of top-down, goal-driven interpretation of "interesting". Instead, we take a bottom-up view using the fact that when people look at an image, their eyes are naturally attracted to certain locations in the image and these locations are fairly consistent across subjects. Thus, our work on visual discovery is closely related to efforts by researchers in biology and psychophysics who attempt to understand and model the human focus of attention process (e.g., [OAE93, IKN98, IK99] and references contained therein).

4.1. Outline of the Approach

Our basic approach is modeled loosely after the human visual system. Images are projected into a visually-relevant subspace using a set of multi-orientation, multi-scale Gabor filters that model the receptive field properties of simple cells in the human visual cortex. Within this filter-response subspace, deviant points are identified through an adaptive statistical test that compares the filter-space description at a given spatial location against a model derived from the local background. Deviant points are spatially agglomerated and grouped across scale and orientation. Details of each step are given in the remainder of this section.

4.2. Filtering

By processing an image with a bank of multi-orientation, multi-scale filters, it is possible to characterize local regions of the image in a "visually relevant" subspace through the filter response values. This approach is motivated by analogy with receptive field properties that have been observed through electrode measurements in the visual cortex of humans and animals [HW62, HW59]. The approach has, in fact, already been used successfully for a number of early vision applications [FA90, GBP$^+$94, JM92, Per95].

There are several filter sets that are in common usage, including Gabor filters, differences of Gaussians (DOG), differences of oriented, offset Gaussians (DOOG), derivatives of Gaussians, and quadratic filters with Gaussian envelopes. In our experiments, we used Gabor filters (see for example [BCG90] and references contained therein) supplemented by center surround and corner-sensitive filters. Each filter responds optimally to features in the image that have a brightness pattern similar to that of the filter.

Given an image $I(x, y)$, each filter f_k is convolved with the image to generate a set of response images $R_k(x, y)$. Similar to the PCA projection described in Section 2, the filtering process can be viewed as a transformation from the low-level pixel representation given by the raw images to a higher-level "feature" representation. The main difference is that here we use a generic set of basis functions, whereas in the recognition work we used an object-specific set of basis functions. The image neighborhood around a given spatial location (x_0, y_0) can now be characterized by the vector

$$\mathbf{R} = [R_1(x_0, y_0), R_2(x_0, y_0), \ldots, R_K(x_0, y_0)]^T \tag{4.6}$$

where K is the number of filters used. Although this representation may be incomplete (i.e., the original image values can not be recovered exactly), the hope is that the visually salient properties will be retained. As one example of the incompleteness of the representation, note that the fine-scale filters only act on the central area of the neighborhood. Any fine-scale detail at the periphery is lost. This limitation could potentially be overcome by including spatially offset filters in the filter bank; however, we have not done so in the current implementation.

4.3. Test for Statistical Deviation

Using the filter response vectors to characterize local regions of the image, we next seek to identify those regions with properties that differ from their surround. We will refer to such regions as *deviants*. For remote sensing imagery, it is expected that localized geological objects such as volcanoes and ice geysers will be identified as deviants (along with a number of other objects/patterns). However, since we make no assumptions about the characteristics of the target objects, except that they are spatially localized, we can only identify deviants based on the fact that they differ from their surround. Statistics of the target class are not available to help in this decision.

Our approach to this subproblem is based on a technique developed for performing target detection using fully-polarimetric radar data [NB90, NBI93]. The basic idea is that detection can be accomplished by *whitening* the clutter and looking for deviations from whiteness. The technique is suboptimal with respect to the situation in which both the clutter and target statistics are known; however, the performance is often close to that of the optimal detector (recognizer).

There are two versions of the whitening filter. In the first version it is assumed that the clutter has a Gaussian distribution with fixed statistics (mean μ and covariance Σ that apply globally across an image). If the filter response vector at a given spatial location is $\mathbf{R}$, then the whitened vector $\mathbf{Z}$ for the same location

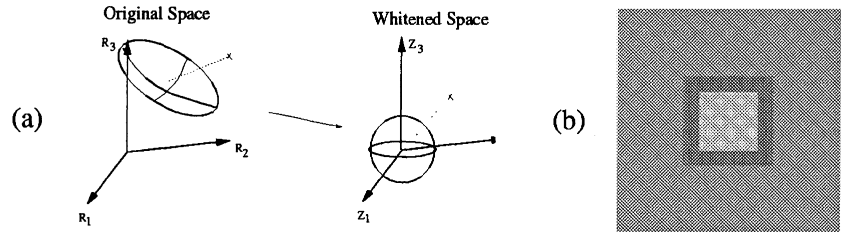


Figure 3. (a) The whitening transformation maps the clutter distribution from the original response space to a new space in which the distribution is isotropic. (b) Stencil used to locally estimate the background distribution. The inner light-colored area corresponds to the size of the filter. The outer dark-grey region is used to statistically characterize the local background. The black area serves as a guard region to insure that the statistical estimates for the background distribution are not contaminated by the central area.

is given by:

$$\mathbf{Z} = \Sigma^{-\frac{1}{2}}(\mathbf{R} - \mu) \tag{4.7}$$

Note that if $\mathbf{R}$ is a random vector from the clutter distribution then $E[\mathbf{Z}] = \mathbf{0}$ and $E[\mathbf{Z}\mathbf{Z}^T] = \mathbf{I}$, where E is the expectation operator and $\mathbf{I}$ is the identity matrix. The scalar variable z given by:

$$z = \sqrt{\mathbf{Z}^T\mathbf{Z}} \tag{4.8}$$

can be used as the basis of a hypothesis test to determine whether $\mathbf{R}$ is from the clutter distribution. Note that z simply corresponds to the Mahalanobis distance of $\mathbf{R}$ from the clutter class. This process can be visualized as shown in Figure 3(a).

A more elaborate version of the whitening filter is locally adaptive, i.e., it assumes that the clutter statistics μ and Σ vary across the image. A vector at a given spatial location can be whitened by first estimating the clutter mean and covariance from nearby regions. In this case, the scalar decision variable is given by:

$$z = \sqrt{(\mathbf{R} - \hat{\mu})^T\hat{\Sigma}^{-1}(\mathbf{R} - \hat{\mu})} \tag{4.9}$$

where $\hat{\mu}$ and $\hat{\Sigma}$ denote locally-generated estimates of the clutter statistics. Typically, these statistics are estimated using a stencil that surrounds the region under test as shown in Figure 3(b). The adaptive whitening filter corresponds to a vector generalization of the classical two-parameter constant false alarm rate (CFAR) algorithm developed by Goldstein for identifying point targets in a clutter background [Gol73].

The whitening approach can be applied to the vector-description of each neighborhood or to each filter response channel individually. For the experiments conducted to date, we have only operated on the channels individually. For a given spatial location, this amounts to estimating a local mean and standard deviation from the nearby filter responses and then calculating how different the location is from the background (in a statistical sense). Treating the channels individually results in fewer parameters that must be estimated from the local background compared to the vector approach which attempts to capture the full joint density[2]. The whitening test is equivalent to evaluating the log-likelihood of the sample given the hypothesized background distribution. The log-likelihood is thresholded to identify interesting points.

4.4. Spatial Agglomeration and Grouping Across Scale

The final step in the algorithm is to try to determine the objects in the image and their approximate sizes by grouping the points obtained from the thresholding step into groups according to their proximity to other detected points. Based on the scale at which a point was identified, all other points within a proportionally-sized radius are agglomerated to form a single group. This action is repeated for each point. The overall procedure is a generalization of the connected components algorithm [Hor86] for a special pseudo-distance measure. Two points A and B are assumed to be from the same class if $|A - B| <= \max(r_A, r_B)$ where r_A and r_B are radii (scales) associated with point A and point B, respectively. This rule is transitively applied so that if A is close enough to B, and B is close enough to C, then A, B, and C are all assigned to the same object even though A and C may not be close (analogous to single-linkage clustering algorithms [DH73]).

4.5. Results

The algorithm was tested on a limited, but diverse set of images collected from various solar system bodies via a variety of imaging techniques. Included in the test set were images of the Moon taken through the ground-based ESO telescope, synthetic aperture radar (SAR) images of Venus collected by the Magellan spacecraft, optical images of Mars collected by Global Surveyor, and optical images of Neptune's moon Triton collected by Voyager II. The goal in this exercise was to determine whether the algorithm could "re-discover" certain localized geological objects that are known to be of interest to planetary scientists without having a specific model of what to look for. As shown in Figure 4, these experiments were largely successful. The system was able to autonomously discover craters on the moon (a), volcanoes on Venus (b), sand

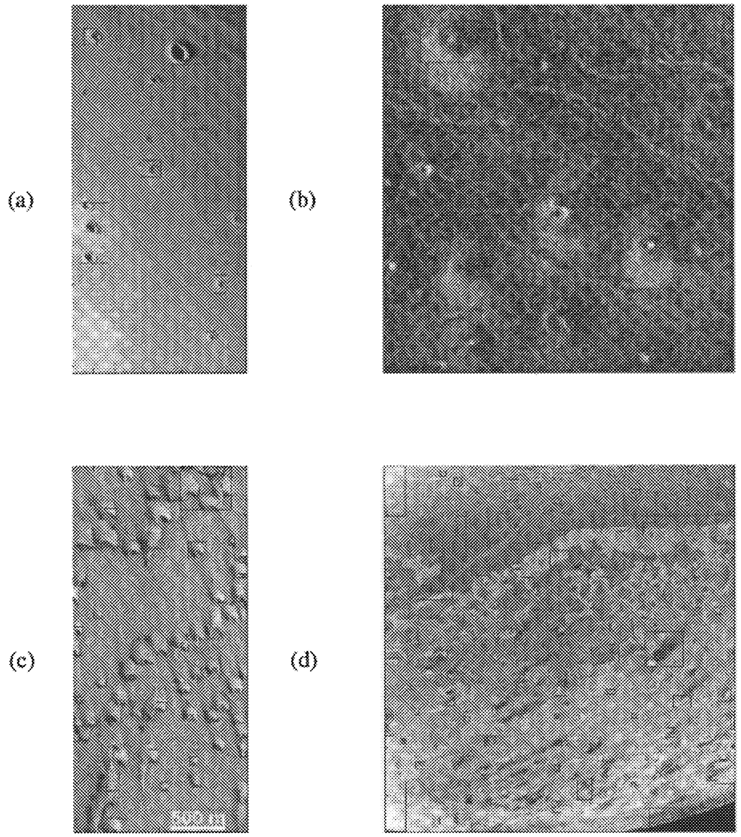
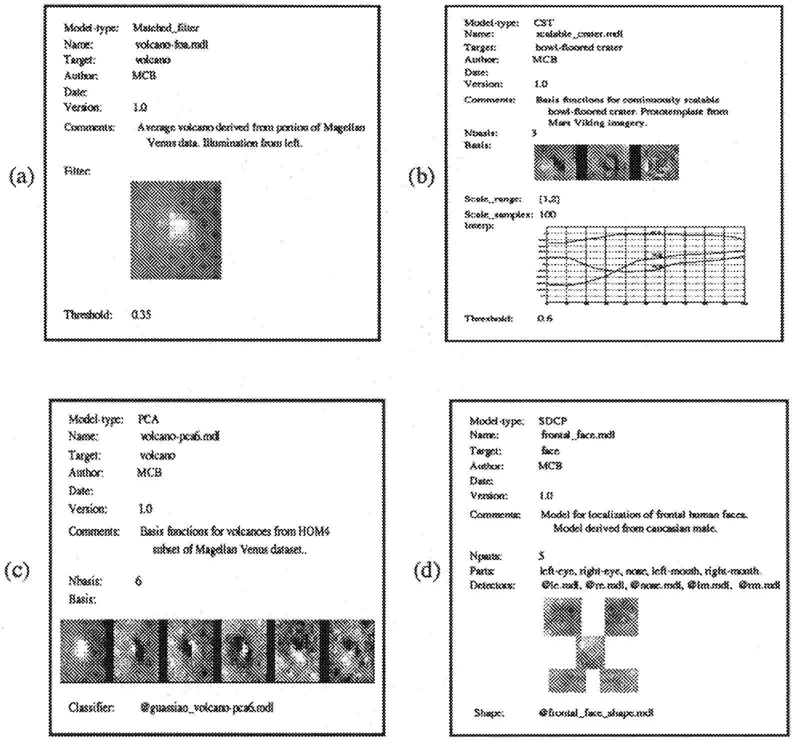


Figure 4. In each subfigure, the regions identified by the discovery algorithm are indicated with boxes. (a) Moon image taken through the ground-based ESO telescope. (b) Venus image collected with synthetic aperture radar by the Magellan spacecraft. (c) Mars image collected by Mars Global Surveyor. (d) Triton image collected by Voyager II spacecraft.

dunes on Mars (c), and ice geysers on Neptune's moon Triton (d). The boxes drawn on the images show the regions that the algorithm identified as interesting. The same algorithm parameters (Gabor filter parameters, thresholds, stencil sizes, etc.) were used in all experiments. The size of each box reflects an estimate of the feature size.

5. Future Directions

As reliable models are developed for recognizing specific objects, the models can be stored in the Diamond Eye system database and tagged with a name



(a)

Model-type: Matched_filter
Name: volcano-fra.mdl
Target: volcano
Author: MCB
Date:
Version: 1.0
Comments: Average volcano derived from portion of Magellan Venus data. Illumination from left.
Filter:
Threshold: 0.35

(b)

Model-type: CST
Name: scalable_crater.mdl
Target: bowl-floored crater
Author: MCB
Date:
Version: 1.0
Comments: Basis functions for continuously scalable bowl-floored crater. Prototemplate from Mars Viking imagery.
Nbasis: 3
Basis:
Scale_range: [1,2]
Scale_samples: 100
Interp:
Threshold: 0.6

(c)

Model-type: PCA
Name: volcano-pca6.mdl
Target: volcano
Author: MCB
Date:
Version: 1.0
Comments: Basis functions for volcanoes from HOM4 subset of Magellan Venus dataset..
Nbasis: 6
Basis:
Classifier: @guassian_volcano-pca6.mdl

(d)

Model-type: SDCP
Name: frontal_face.mdl
Target: face
Author: MCB
Date:
Version: 1.0
Comments: Model for localization of frontal human faces. Model derived from caucasian male.
Nparts: 5
Parts: left-eye, right-eye, nose, left-mouth, right-mouth.
Detectors: @le.mdl, @re.mdl, @nose.mdl, @lm.mdl, @rm.mdl
Shape: @frontal_face_shape.mdl

Figure 5. Recognizers and query models are accumulated and stored in the system database along with semantic information that will enable users to retrieve models by name. (a) Matched filter for Venusian volcanoes, (b) Continuously-scalable template model for a particular subclass of impact craters, (c) PCA model for Venusian volcanoes, (d) SDCP model for human faces.

according to the type of target object. Figure 5 shows a cartoon representation of several recognizer/query models. A user, who is preparing to search for some type of object, such as craters, can first check the database to determine if such a model already exists. As the user base grows and more models are contributed to the database, users will be able to search for many different types of objects simply by specifying the name of the object. The system will pull the appropriate recognizer/query model from the database and apply the model to one or more images as requested by the user. In the future, users will have the ability to rate models along different dimensions (speed, accuracy, etc.) and provide feedback that can be used to refine a model (e.g., the user may say that the following identifications produced by crater model xyz do not appear to be craters).

6. Summary

Mining useful content from image collections is very much an interdisciplinary endeavor that draws upon expertise in computer vision, image understanding, data mining, machine learning, databases, distributed/parallel computing, software design, and artificial intelligence. Given the ever-increasing size of datasets, automated techniques must be developed and employed if more than a fraction of the data is to be examined. One source of encouragement is that we know humans can extract meaningful information from images, In contrast, in many "pure" data mining tasks it is not clear whether useful knowledge can be obtained from a mountain of abstract tables.

Diamond Eye is a distributed software architecture that is being developed to enable users (scientists) to analyze large image collections by interacting with one or more custom data mining servers via a Java applet interface. Diamond Eye provides an application-independent infrastructure into which domain-specific algorithms can be easily added. The architecture and algorithms described herein highlight our efforts to develop a large-scale image mining system for extracting knowledge from NASA's data archives.

Acknowledgments

The research described in this article has been carried out in part by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

The JARtool work was carried out in collaboration with P. Smyth, U. Fayyad, P. Perona, L. Crumpler, J. Aubele, L. Asker, and M.P. Burl. The Diamond Eye work was was carried out in collaboration with C. Fowlkes, J. Roden, A. Stechert, and S. Mukhtar. Planetary scientists W. Colwell, W. Merline, C. Chapman, and A. Stern of the Southwest Research Institute have served as a scientific advisory panel on the Diamond Eye work. Contributions to the query and discovery algorithms were made by C. Fowlkes, D. Lucchetti, T. Stough, and T. Mann. Portions of this work have previously appeared in [BAS+98, BFR+99b, BFR99a, BL00].

C. Fowlkes and D. Lucchetti participated in these projects under the umbrella of the Summer Undergraduate Research Fellowship (SURF) program administered through the Office of Student-Faculty Programs at the California Institute of Technology.

Notes

1. We have also experimented with clustering the training data in pixel space and applying PCA separately to each cluster. This extension yields an approximation to the manifold by a union of hyperplanes. See also recent results on nonlinear PCA [RS00].

[2] We note that there is also a relatively new technique for texture analysis that uses independent components analysis [MP99] to find channels (linear combinations of the basic Gabor filter channels) such that a product of scalar densities over the channels approximates the joint density as closely as possible.

References

[AM95] Y. Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272(4):64–69, April 1995.

[AM97] L. Asker and R. Maclin. Achieving expert performance on real-world problems using machine learning. In *Fourteenth International Conference on Machine Learning*, 1997.

[AS90] J.C. Aubele and E.N. Slyuta. "Small Domes on Venus: Characteristics and Origins". *Earth, Moon, and Planets*, 50/51:493–532, 1990.

[Bal94] P. Baldi, 1994. private communication.

[BAS+98] M.C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler, and J. Aubele. Learning to recognize volcanoes on Venus. *Machine Learning*, 30:165–194, 1998.

[BB98] M. Bennamoun and A. Bodnarova. Automated visual inspection and flaw detection in textile materials: Past, present, and future. In *IEEE Int. Conf. on Systems, Man, and Cybernetics*, volume 5, pages 4340–4343, 1998.

[BCG90] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 12(1):55–73, January 1990.

[BFP+94] M.C. Burl, U.M. Fayyad, P. Perona, P. Smyth, and M.P. Burl. Automating the hunt for volcanoes on Venus. In *Proc. of the Computer Vision and Pattern Recognition Conference*, pages 302–309, 1994.

[BFPS96] M.C. Burl, U.M. Fayyad, P. Perona, and P. Smyth. Trainable cataloging for digital image libraries with applications to volcano detection. Technical report, Technical Report CNS-TR-96-01, California Institute of Technology, Dept. of CNS, 1996.

[BFR99a] M.C. Burl, C. Fowlkes, and J. Roden. Mining for image content. In *Systemics, Cybernetics, and Informatics / Information Systems: Analysis and Synthesis*, July 1999.

[BFR+99b] M.C. Burl, C. Fowlkes, J. Roden, A. Stechert, and S. Mukhtar. Diamond Eye: A distributed architecture for image data mining. In *SPIE Aerosense Symposium, Conf. on Data Mining and Knowledge Discovery*, 1999.

[BL00] M.C. Burl and D. Lucchetti. Autonomous visual discovery. In *SPIE Aerosense Symposium, Conf. on Data Mining and Knowledge Discovery*, 2000.

[BLP95] M.C. Burl, T.K. Leung, and P. Perona. "Face Localization via Shape Statistics". In *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995.

[BLP96] M.C. Burl, T.K. Leung, and P. Perona. "Recognition of Planar Object Classes". In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 1996.

[BWP98] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *European Conf. on Computer Vision*, 1998.

[CFA91] F.S. Cohen, Z. Fan, and S. Attali. Automated inspection of textile fabrics using textural models. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 13(8):803–808, August 1991.

[Che96] K. Cherkauer, 1996. private communication.

[DB00] D. DeCoste and M.C. Burl. Achieving distortion-invariant recognition via jittered queries. In *Proc. of Computer Vision and Pattern Recognition Conf.*, 2000.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc., 1973.

[FA90] W. Freeman and E. Adelson. Steerable filters for early vision, image analysis and wavelet decomposition. In *Third International Conference on Computer Vision*, pages 406–415. IEEE Computer Society, 1990.

[FA91] W. Freeman and E Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:891–906, 1991.

[FMW97] D. Forsyth, J. Malik, and R. Wilensky. Searching for digital pictures. *Scientific American*, pages 88–93, June 1997.

[FSN$^+$95] M. Flickner, H Sawhney, W Niblack, et al. "Query by Image and Video Content – The QBIC System". *Computer*, 28(9):23–32, 1995.

[GBP$^+$94] H. Greenspan, S. Belongie, P. Perona, R. Goodman, S. Rakshit, and C. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, Seattle, June 1994.

[Gol73] Goldstein. False alarm regulation in Weibull and log-normal clutter. *IEEE Trans. on Aerospace and Electronics Systems (AES)*, 9(1):84–92, January 1973.

[Hor86] B.K.P. Horn. *Robot Vision.* MIT press, 1986.

[HW59] D. Hubel and T. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol. (Lond.)*, 148:574–591, 1959.

[HW62] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol. (Lond.)*, 160:106–154, 1962.

[IK99] L. Itti and C. Koch. Learning to detect salient objects in natural scenes using visual attention. In *Image Understanding Workshop*, 1999.

[IKN98] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 20(11):1254–1259, 1998.

[JM92] D. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proc. 2^{nd} Europ. Conf. Comput. Vision, G. Sandini (Ed.), LNCS-Series Vol. 588, Springer-Verlag*, pages 395–410, 1992.

[LBP95] T.K. Leung, M.C. Burl, and P. Perona. "Finding Faces in Cluttered Scenes". In *Intl. Conf. on Computer Vision*, Cambridge, MA, 1995.

[LBP98] T.K. Leung, M.C. Burl, and P. Perona. Probabilistic affine invariants for recognition. In *Proc. of the Computer Vision and Pattern Recognition Conference*, 1998.

[ME00] Michael C. Malin and Kenneth S. Edgett. Evidence for recent groundwater seepage and surface runoff on mars. *Science*, 288:2330–2335, June 30, 2000.

[MP99] R. Manduchi and J. Portilla. 'independent component analysis of textures. In *Seventh Intl. Conf. on Computer Vision*, pages 1054–1060, 1999.

[MZ92] J.L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision.* Artificial Intelligence. The MIT Press, 1992.

[NB90] L.M. Novak and M.C. Burl. Optimal speckle reduction in polarimetric sar imagery. *IEEE Trans. on AES*, 26(2):293–305, March, 1990.

[NBI93] L.M. Novak, M.C. Burl, and W.W. Irving. Optimal polarimetric processing for enhanced target detection. *IEEE Trans. on AES*, 29(1):234–244, January, 1993.

[NBM96] S. Nayar, S. Baker, and H. Murase. Parametric feature detection. In *Proc. of the Computer Vision and Pattern Recognition Conference*, pages 471–477, 1996.

[OAE93] B.A. Olshausen, C.H. Anderson, and D.C. Van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *J. Neuroscience*, 13(11):4700–4719, November 1993.

[Per95] P. Perona. Deformable kernels for early vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(5):488–499, 1995.

[PFJ+91] G.H. Pettengill, P.G. Ford, W.T.K. Johnson, R.K. Raney, and L.A. Soderblom. "Magellan: Radar Performance and Data Products". *Science*, 252:260–265, 1991.

[PP96] R.W. Picard and A.P. Pentland. Introduction to the special section on digital libraries: Representation and retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(8):769–853, Aug 1996.

[RH95] I. Rigoutsos and R. Hummel. A bayesian approach to model-matching with geometric hashing. *Computer Vision and Image Understanding*, 62(1):11–26, 1995.

[RS00] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 22, 2000.

[SB90] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *J. of the American Society for Information Science*, 41(4):288–297, 1990.

[SB97] T. Stough and C. Brodley. Image feature reduction through spoiling: its application to multiple matched filters for focus of attention. In *Proc. of the Third Annual Conference on Knowledge Discovery and Data Mining*, pages 255–259, 1997.

[SSA+92] R.S. Saunders, A.J. Spear, P.C. Allin, R.S. Austin, A.L. Berman, R.C. Chandlee, J. Clark, A.V. Decharon, and E.M. Dejong. Magellan mission summary. *Journal of Geophysical Research Planets*, 97(E8):13067–13090, 1992.

[SSG99] H. Sari-Sarraf and J.S. Goddard Jr. Vision system for on-loom fabric inspection. *IEEE Trans. on Industry Applications*, 35(6):1252–1259, 1999.

[SYD93] P. Simard, Y.L.Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5*, pages 50–58, 1993.

[Tur96] M. Turmon, 1996. private communication.

[Vir] Virage. Visual image retrieval engine. URL: http://206.169.1.82/market/vir.html.

[Wol90] H.J. Wolfson. Model-based object recognition by geometric hashing. In *Proc. 1^{st} Europ. Conf. Comput. Vision, LNCS-Series Vol. 427, Springer-Verlag*, pages 526–536, 1990.

Chapter 5

MINING ASTRONOMICAL DATABASES

Roberta M. Humphreys, Juan E. Cabanela, and Jeffrey Kriessler

Abstract The development of software tools and techniques for the efficient access and analysis of large astronomical databases poses some unique challenges. We briefly describe some of the problems astronomical data and datasets present and give an example from our own efforts to automate the classification of galaxies, and then discuss where "clustering" algorithms may be applicable.

Keywords: databases galaxy, morphologiacal classification, data mining

1. Introduction

The number of space-based all-sky surveys ranging from gamma rays and X-rays to the far infrared and millimeter wavelengths is rapidly increasing. When we add in the digitization of the historical photographic surveys like the Palomar Observatory Sky Survey (POSS I and II) and the new ground-based digital surveys, like 2MASS (Two Micron Astronomical Sky Survey) and the optical Sloan Digital Sky Survey (SDSS), a $\geq$40 TB-sized Internet-wide multi-wavelength astronomical dataset will soon exist. To derive the maximum scientific benefit from this vast resource of fundamental data and the recently proposed National Virtual Observatory will require efficient access, such as federated databases that stretch across several databases at physically different locations, and new software techniques and tools, often referred to as "data mining", for the analysis of large databases.

Astronomical databases, however, pose unique problems due not only to their very large size, but also to the variable quality of the data and the uncertainty of measurement over the entire electromagnetic spectrum. The nature of astronomical objects with their very wide dynamic range in apparent luminosity and in apparent size (angular diameter) present

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 85–94.
© 2001 *Kluwer Academic Publishers.*

additional challenges. Astronomical databases will not only possess an unprecedented number of objects, but the astronomical objects themselves may also have a large number of attributes leading to a very high dimensional dataset.

Many of the necessary techiques and software packages, including artificial intelligence techniques, like neural networks and decision trees have already been successfully applied to astronomical problems such as pattern recognition and object classification, while new clustering and data association algorithms that may have application to large astronomical databases are being developed by computer science groups. However, these new software packages are often developed and tested on idealized or "clean" datasets that lack the "real noise" and uncertainty of measurement encountered particularly in large astronomical databases. The APS Catalog of the POSS I is an excellent resource for perfecting and testing these data mining techniques.

2. The APS Project

The University of Minnesota's Automated Plate Scanner(APS) Catalog of the POSS I is an on-line database of fundamental data and parameters for over 100 million stars and galaxies derived from our digitized scans of glass copies of the blue and red plates of the original, first epoch Palomar Observatory Sky Survey (POSS I). It is large enough, 25 GB, to present a realistic challenge for testing "data mining" algorithms on a range of astrophysical applications. Its scientific usefulness and validity have been demonstrated by numerous studies by members of our APS group and the astronomical community who have used the database for a variety of research applications. For examples, see [OA95, LH94, LH96] and [Whi99]).

The Catalog contains coordinates, magnitudes, colors, and several other computed image parameters for all of the matched images on the blue and red plates. It provides information for the individual stars and galaxies down to fainter than 21st magnitude (in the blue). The calculation of accurate image parameters and the reliable separation of stellar and non-stellar images (galaxies) has long been a focus of our work with the APS data. We were the first group to successfully apply AI techniques, specifically neural networks, to the image classification problem [Ode92, Ode93, Ode95]. Our neural network image classifier has been trained to the faint limit of the photographic plates and gives a success rate better than 90% to within one magnitude of the plate limit. It uses various image parameters with a back-propagation algorithm and two hidden layers to generate an output layer with two nodes, star or non–star(= galaxy). This "node_gal" value ranging from 0 to 1 also provides a confidence level of the classification and is cataloged with the image type.

The completed catalog of objects is available as an on-line database over the Internet (URL is http://aps.umn.edu). Querying is achieved with a custom-designed database management system called StarBase capable of handling millions of entries. StarBase was developed in collaboration with faculty and students of the University of Minnesota Computer Science Department. It uses specialized hashing on each image parameter derived for every catalog entry, including a two-dimensional hierarchical algorithm for positional search and retrieval. This level of optimization provides us with a DBMS that is faster and smaller than a commercial equivalent. A complementary image database is also available and includes all of the matched images in the object catalog as well as the unmatched images above the noise threshold on both the blue and red plates.

We have recently installed a federated database (FDBS) called Myriad [LHS95] developed by Professor Jaideep Srivastava's group in our Computer Science Dept. The FDBS integrates the APS catalog and image database so that they appear as one easy-to-use resource. With a FDBS, the queries and transactions on the integrated database are performed as if it were a single database. The separate DBMS's are hidden from view by a flexible interface. It is important to emphasize that the FDBS permits horizontal access to the data, not just vertical. Queries can be made not only by sky position, but also by any parameter in either database.

Although we have had considerable success with our neural network-based object classifier for our research applications, for many astrophysical problems the actual morphological type of the galaxy is very important, especially for studies of galaxy formation and evolution and large-scale structure in the universe. Working with members of our Computer Science Department, we have recently had some success, described in the next section, applying data mining and pattern recognition codes to identifying the most useful parameters for automating the classification of the galaxy images by their morphological types.

3. The Morphological Classification of Galaxies

Ever since the discovery of galaxies, it has been known that these assemblies of stars, dust and gas have different morphological shapes. In 1936, Edwin Hubble established a system to classify galaxies into three fundamental types. Elliptical galaxies had an elliptical shape with no other discernible structure. Spiral galaxies had an elliptical nucleus surrounded by a flattened disk of stars and dust containing a spiral pattern of brighter stars. The irregular galaxies as their name suggests, were irregularly shaped and did not fit into the other two categories. As more galaxies were observed, it became apparent that the galaxy types formed a continuous sequence starting from nearly spherical galaxies toward more flattened ellipticals, through the lenticulars (galaxies with a large nucleus and small disk with no spiral structure) to the true spirals starting with tightly wound spiral arms and proceeding to less tightly wound arms and concluding with the irregulars. In other words, Hubble arranged galaxies in order of increasing complexity. Although many subdivisions and refinements have been made within the Hubble classification system, we are primarily concerned here with identifying the four basic types of galaxies: ellipticals, lenticulars, spirals, and irregulars.

The classification of galaxies is typically performed by visual inspection of photographic plates. This is by no means an easy task, requiring a great deal of practice and time on the part of the astronomer. Large catalogs of galaxies containing a few tens of thousands of galaxies [e.g. the Third Reference Catalogue of Bright Galaxies [deV91] take years to compile. With today's large all–sky surveys, generating millions of galaxy images, human classification is no longer a viable option. Furthermore, although the types are well defined, human classifications tend to be subjective and it is difficult for independent researchers to reproduce results. Often it is very difficult to distinguish between adjacent types. For instance, a lenticular galaxy viewed face-on, looking down at the disk, looks very similar to an elliptical and all morphological catalogs have fewer face-on lenticulars than edge-on, where the presence of a disk is more readily discerned. Studies also show that morphological catalogs of galaxies produced by even the best human classifiers disagree with other classifiers between 10% and 20% of the time. Therefore, in order to produce the large, objective, reproducible, morphological catalogs necessary for galaxy formation and evolution studies, computer generated classifications are required.

There have been a few recent attempts to create an automated classification system, generally using artificial neural networks [Ode95, NLS95]. While limited success has been achieved, these computer-based classifiers have yet to produce large, unbiased morphological galaxy catalogs. The reason for this seems to be that while it has been possible to train a neural network to correctly classify a well defined, hand picked, set of galaxies, when applied to the large random samples of galaxy images upon which any classifier must ultimately be applied, they fail to give results that can equal human classifications.

In our attempts to solve this problem we have visually classified some 1500 galaxy images obtained from the APS database in the region of the north galactic pole. This training set was chosen based solely on the brightness and size of galaxy images on 9 photographic plates. Images that were difficult to classify (less than 1 %) were removed from this sample. The first problem is to identify a set of parameters which can separate the galaxies by their types. This has turned out to be quite challenging. The human eye can easily recognize complicated patterns in images such as spiral arms which tend to be spotty, blotchy affairs that are difficult for automated techniques. Often it is necessary to rely on secondary effects such as color (spiral galaxies tend to be bluer than ellipticals) which are not specifically part of the classification system as originally conceived. If a picture is worth a thousand words, with a little imagination a galaxy image can be described by hundreds of parameters, all of which may have some relation to the morphological type. Currently, we calculate over five hundred such parameters for each galaxy in the APS database. Unfortunately, we cannot simply present all of these parameters to a neural network and let the training algorithm determine which are the most important. We would merely end up with a network that has memorized the training sample perfectly, but performs poorly on samples not seen during training. In order to have a reasonable chance of spanning a five hundred dimensional parameter space we would require a training sample of many millions: the thing we are trying to avoid.

With drastic increases in training set size ruled out for practical reasons, another option is to limit the number of parameters presented to a neural network. The question is, which parameters to choose? If one or two parameters yielded adequate separation, we could merely plot all the parameters in turn and see which provided the greatest distance between the clumps defining the various types. Unfortunately, this is not the case. While several parameters show trends with galaxy type, no combination of two or three parameters is capable of solving the problem.

The problem of finding clusters in large dimensional spaces however, falls within the sphere of data mining. Working with the data mining group at the University of Minnesota, we applied the program Mineset to the task. This program allows quick evaluation and ranking of the parameters, as well as creating a decision tree classifier. Using the 10 best parameters from the column importance tool in Mineset, we have been able to achieve a classifier with an 85% accuracy treating the ellipticals and the lenticulars as a single class. While this is still short of our goal of creating an automated classifier as good as the human classifiers, it is a step in the right direction. In order to improve our classifier we continue to seek parameters that can provide better separation between the morphological types. However, it is possible that our 500 parameters already have enough information to correctly classify galaxies and by limiting the number to only ten we are ignoring useful information. At the same time, examination of the misclassified galaxies often reveals an anomaly in the image which confuses the computer classifier. Examples include foreground stars or faint background galaxies superimposed on the galaxy image or the presence of dust lanes in an otherwise structureless elliptical galaxy. While a human classifier routinely discounts these deviations, automated classifiers see only the parameters presented to them. A possible solution is to train a large number of neural networks, each of which is presented a small number of the parameters. The final classification is then taken to be a weighted average of all the classifiers output. This procedure would permit a more robust classification to be performed as one or two deviant parameters can be out-weighed by the vast majority of normal parameters for that galaxy type.

4. Testing Clustering Algorithms on an Astronomical Databse

To a computer scientist, "clustering" is a discovery process [SAD93, CHV96] that groups objects such that the similarity between objects in the same group is maximized and the similarity between objects in different groups is minimized [JD88, KR90, CHV96]. In astronomy, we often group objects into "populations" with distinct properties. There is obviously a large overlap between what an astronomer would call a "population" and what a computer scientist would call a "cluster." In this section we will concentrate on explaining how new clustering algorithms can be applied to existing (and future) astronomical databases to allow for automated identification of astronomical populations.

There are many different types of astronomical populations. If the properties distinguishing the populations are spatial (positions on the sky or in space), the populations identified may be real physical "clusters" of objects. For example, it has been well established that galaxies tend to lie near other galaxies, in "galaxy clusters." Spatial clusters are one very common form of population in astronomy. However, in addition to these spatial clusters, populations with similar physical and image parameters may exist both within the spatial clusters and independent of them. An example of a non-spatial population is the famous Hertzsprung-Russell diagram, a plot of color versus absolute brightness for stars. Stars tend to lie in relatively restricted regions in this two–dimensional parameter space. The stars in this diagram can therefore be classified into separate populations, called main sequence stars, giants, white dwarfs, etc. These stellar populations have other properties, outside of color and absolute brightness, that distinguish them, indicating they are indeed physically distinct types of stars. If we treat spatial, physical, and image parameters as part of a multivariate description of each object in an astronomical database, we can see that astronomical populations are just what a computer scientist would call a "cluster."

Traditionally the identification of astronomical populations has been done "by eye" through examination of parameter space plots. Recently, astronomers have started using computers to identify some populations in astronomical databases, usually by looking for populations they expect to exist. The promise of applying clustering algorithms to astronomical databases lies in the application of precisely defined criteria in identifying populations, criteria that are not subject to psychological or physiological biases. Developing clustering algorithms for astronomical datasets poses a number of challenges due to both the characteristics of the data discussed as well as the types of the desired clusters. The clusters may be of variable sizes and densities, and of arbitrary shapes. We need to develop clustering algorithms that are capable of accommodating different clustering objectives.

For spatial clustering, a new class of hierarchical agglomerative clustering algorithms are useful in identifying clusters with varying, non-uniform densities and arbitrary shapes. These algorithms use a dynamic-modeling approach to measure the similarity between two clusters; thus allowing them to automatically adjust to the characteristics of each cluster. Two clusters are merged only when the distinctiveness of parameter values between the clusters is comparable to the internal scatter of the parameter values within each cluster. These algorithms forego the definition of a user-specified model (and the biases possible in such models), instead automatically adjusting the parameters distinguishing clusters from one another. Therefore, these algorithms are suitable for identification of spatial clusters of stars and galaxies, since they can identify clusters with homogeneous internal parameters even if the clusters vary in density, shape, or size.

An example of such a clustering algorithm is Chameleon [KAK99]. Chameleon operates on a sparse graph where nodes represent objects and weighted edges represent the similarities between objects. This sparse graph representation allows Chameleon to scale to the large data sets becoming common in astronomy. Initial investigations with Chameleon show that it is able to correctly identify clusters of varying size, orientation, shape. In addition to this strength, Chameleon is tolerant of noise and outliers, something common to all astronomical datasets. It is our goal to apply Chameleon to a subset of the APS catalog, a catalog of over 200,000 galaxies in the region surrounding the North Galactic Pole, called the MAPS-NGP. Using Chameleon on the MAPS-NGP, we hope to identify not only previously recognized galaxy clusters [Abe58], but new ones as well. The challenge when applying such an algorithm to astronomical datasets is that every object will usually have a great variety of parameters describing it, thus essentially, instead of a sparse graph, we have a dense graph.

Unsupervised clustering can also be used to discover structure within a large dataset as well as grouping similar objects together independent of any user-defined classes. Once such structuring of the data is preformed, further exploration of the data set is made easier, since analyses that would normally have been applied to the entire data set can now be applied to hierarchically structured subsets. The Principal Direction Divisive Partitioning (PDDP) algorithm has been demonstrated to be a fast method of constructing a hierarchical clustering tree top-down [Bol98, CHV96]. PDDP is fast and scalable to very large datasets, producing data structures that identify the attributes distinguishing one cluster from another, in a completely unsupervised way. Initial application of the PDDP algorithm to the MAPS-NGP should provide a way of identifying distinct populations of objects, with information on just what makes them distinct.

We have found that the problems with existing data mining techniques when applied to astronomical datasets are largely tied to the vast size of modern astronomical datasets plus the density of records. Often, records for an individual astronomical object will contain dozens of fields (each with a value). And in modern, multi-wavelength astronomy, cross-identification of the same object in separate databases is a common technique. See [CD01] for an example of cross-identified radio and optical data used to identify a subset population of low surface brightness galaxies from the general galaxy population. Data mining techniques will be required to handle hundreds of fields associated with each record, with the values of items in those fields retrieved through integrated and efficient access to multiple, distributed, databases around the world. When data mining techniques meet this challenge, the process of astronomical discovery should accelerate tremendously, and the astronomer's efforts will be able to focus on interpreting the relationships for astronomical populations (maybe we'll use the term "cluster" by then) instead of searching for them.

References

[Abe58] G. O. Abell. The Distribution of Rich Clusters of Galaxies. Astrophys. J. Suppl., 3, 211, 1958.

[Bol98] D. Boley. Principal Direction Divisive Partitioning. Data Mining and Knowledge Discovery, 2, 325, 1998.

[CD01] J. E. Cabanela and J. M. Dickey. Galaxies on the Blue Edge. Astron. J., to be published, 2001.

[CHV96] M. S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from a Database Perspective. In IEEE Transactions on Knowledge and Data Engineering, 8(6), 866, 1996.

[deV91] G. de Vaucouleurs, A. de Vaucouleurs, H. G. Corwin, Jr., R. J. Buta, G. Paturel, and P. Fouque. Third Reference Catalogue of Bright Galaxies (Springer-Verlag New York Inc., New York), 1991.

[HBG98] S.Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. A Web Agent for Document Categorization and Exploration. In Autonomous Agents '98 Conf.

[JD88] A. K. Jain and R. C. Dubes. Algorithms for Clustering Data, Prentice Hall, 1988.

[KAK99] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel Hypergraph Partitioning: Applications in VLSI Domain. In IEEE Transactions of VLSI Systems, 7(1), 69, 1999.

[KR90] L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley and Sons, 1990.

[LH94] J. A. Larsen and R. M. Humphreys. The Axial Ratio of the Galactic Stellar Halo. Astrophys. J. (Letters), 436, L149, 1994.

[LH96] J. A. Larsen and R. M. Humphreys. A Large Asymmetry in the Galactic Distribution of Halo/Thick Disk Stars. Astrophys. J. (Letters), 468, L99, 1996.

[LHS95] E.-P. Lim, S.-Y. Hwang, J. Srivastava, D. Clements, and M. Ganesh. Software Practice and Experience, 25, 533, 1995.

[NLS95] A. Naim, O. Lahav, L. Sodre, and M. C. Storrie-Lombardi. Automated Morphological Classification of APM Galaxies by Supervised Artificial Neural Networks. Mon. Not. Roy. Astron. Soc., 275, 567, 1995.

[Ode95] S. C. Odewahn. Automated Classification of Astronomical Images. Pub. Astron. Soc. Pacific, 107, 770, 1995.

[OA95] S. C. Odewahn and G. Aldering. Galaxy Properties at the North Galactic Pole. Astron. J., 110, 2009, 1995.

[Ode92] S. C. Odewahn, E. B. Stockwell, R. L. Pennington, R. M. Humphreys, and W. Zumach. Automated Star–Galaxy Discrimination with Neural Networks. Astron. J., 103, 318, 1992.

[Ode93] S. C. Odewahn, R. M. Humphreys, G. Aldering, and P. M. Thurmes. Star–Galaxy Separation with a Neural Network II. Multiple Schmidt Plates. Pub. Astron. Soc. Pacific, 105, 1354, 1993.

[SAD93] M. Stonebraker, R. Agrawal, U. Dayal, E. J. Neuhold, and A. Reuter. DBMS Research at a Crossroads: The Vienna Update. In Proc. Of the 19th VLDB Conference, 1993.

[Whi99] R. L. White, *et al.*. The FIRST Bright Quasar Survey II. Astrophys. J. Suppl., 126, 133, 1999.

Chapter 6

SEARCHING FOR BENT-DOUBLE GALAXIES IN THE FIRST SURVEY

Chandrika Kamath, Erick Cantú-Paz, Imola K. Fodor, and Nu Ai Tang

Abstract Data mining techniques are increasingly gaining popularity in various scientific domains as viable approaches to the analysis of massive data sets. In this chapter, we describe our experiences in applying data mining to a problem in astronomy, namely, the identification of radio-emitting galaxies with a bent-double morphology. Until recently, astronomers associated with the FIRST (Faint Images of the Radio Sky at Twenty-cm) survey identified these galaxies through a visual inspection of images. While this manual approach has been very subjective and tedious, it is also becoming increasingly infeasible as the survey has grown in size. Upon completion, FIRST will include almost a million galaxies, making the use of semi-automated analysis methods necessary. We describe the FIRST data set and the problem of identifying bent-double galaxies. We discuss our solution approach, focusing on the challenges we face in the application of data mining to a scientific data set. We explain why, in contrast with most commercial data mining applications, data preprocessing requires a considerable effort in scientific applications. Using decision tree classifiers, we describe the work we are doing in the detection of bent-double galaxies. Our results indicate that data mining techniques, steered by proper domain knowledge, can greatly enhance the manual exploration of massive data sets.

Keywords: Data mining, astronomical surveys, radio-emitting galaxies, role of data preprocessing

1. Introduction

Data mining is a process concerned with uncovering patterns, associations, anomalies, and statistically significant structures and events in data ([KM00] and the references therein). It is an iterative and interactive process involving data preprocessing, search for patterns, and interpretation of the results. While there is broad consensus on what constitutes data mining, the tasks that are performed in each step depend on the problem domain, the problem being solved,

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 95–114.

© 2001 *Kluwer Academic Publishers.*

and the data itself. As explained later, scientific datasets present particular challenges that require a careful tailoring of the data mining tasks to the problem and dataset.

In this chapter, we describe how the Sapphire project [Sap] has defined the tasks in data mining to make the process more suitable for addressing the diverse needs of scientific and engineering applications. We discuss our work in the context of one of the data sets we are analyzing from astronomy, where we are interested in identifying radio-emitting galaxies with a bent-double morphology. Our goal in this effort is to replace the current visual inspection of images by a semi-automated, and hopefully more objective, approach based on data mining techniques. We discuss the challenges we have faced in trying to reach this goal, and the approach we have taken to address some of these problems. In particular, we show that data preprocessing is not only crucial to the successful application of data mining, but can be a difficult and time consuming task for scientific data.

This Chapter is organized as follows: Section 2 describes our view of scientific data mining, focusing in particular on the tasks that are specific to scientific and engineering data sets. Section 3 describes our astronomy data set, namely the FIRST survey, and outlines the problem of detecting bent-double radio-emitting galaxies. Section 4 provides details on the approach we have taken to mine the FIRST data set for bent-doubles, and the difficulties we have faced in the process. Section 5 reports our results, focusing on the important role played by the data preprocessing step in data mining. Section 6 concludes with our observations, the lessons we have learned, and plans for future work.

2. Scientific Data Mining

Data sets that arise in various physical sciences and engineering applications are usually the result of computer simulations, observations, or experiments. These data are typically in the form of signals (including images) or mesh data. Often, one or more types of data may be available for a problem, in which case data fusion is required in order to exploit all the information and make a more accurate decision.

In light of these observations, our definition of data mining starts with the raw data and includes extensive preprocessing (Figure 1). If the raw data is very large, we may use sampling and work with fewer instances, or use multiresolution techniques and work with data at a coarser resolution. This first step may also include data fusion, if required. Next, noise is removed and relevant features that represent and describe the items of interest are extracted from the data. These features must not only be robust, that is, insensitive to small changes in the data, but also be invariant to scaling, translation, and rotation. At the end of this step, we have a vector of features for each data item. These

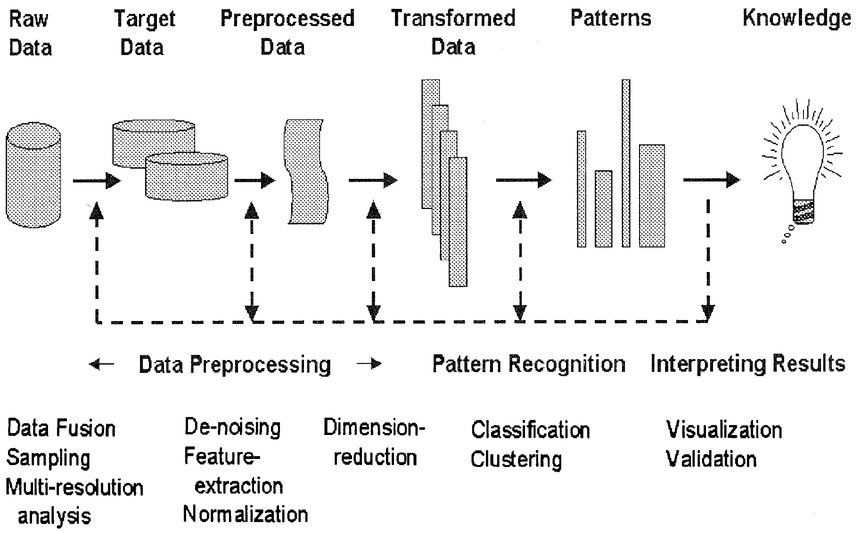

Raw Data
Target Data
Preprocessed Data
Transformed Data
Patterns
Knowledge
← Data Preprocessing →
Pattern Recognition
Interpreting Results
Data Fusion
Sampling
Multi-resolution analysis
De-noising
Feature-extraction
Normalization
Dimension-reduction
Classification
Clustering
Visualization
Validation

Figure 1. Data mining: an iterative and interactive process.

vectors are normalized to remove any skew resulting from possibly different units. Next, depending on the problem and the data, we may need to reduce the number of features using dimension reduction techniques such as principal component analysis (PCA) or its non-linear versions. After this preprocessing, the data is ready for the detection of patterns, through the use of traditional techniques such as classification and clustering. The patterns obtained are then displayed to the domain scientist for validation.

The data mining process is iterative and interactive; any step may lead to a refinement of one or more of the previous steps. To ensure the success of data mining, the domain scientist should be actively involved in all stages, starting from the initial description of the data and the problem, the identification of potentially relevant features and the training set (where necessary), and the validation of the results.

It must be noted that several of the tasks in scientific data mining are independent of the data or the problem, while others are more specific to the domain. However, for scientific data in the form of signals, images, or meshes, there is ample opportunity for re-use of algorithms and software. For example, while wavelet-based de-noising techniques can be used in general to de-noise signals and images, the particular type of wavelet used might depend on the statistical properties of the data and noise, which in turn is determined by the sensors that were used to collect the data. Similarly, while decision trees may be used in the classification of both astronomy and simulation data, it may turn out that neural networks are better at classification for a particular problem in astronomy. In addition, techniques that may have been developed for one kind of data, such as segmentation for images, can be applied to a similar task in another kind of data, such as segmentation in mesh data. However, tasks such as reading, writing, and displaying data are typically very specific to the format used for the data.

As part of the Sapphire project, we are developing toolkits for each of the computationally intensive modules described above. This object-oriented framework, written in C++, is designed to be modular, with the ability to handle both signal and mesh data types. Further details on this project can be found in [Sap, KCP00, KBFT00].

We next describe how we are applying the data mining process described in this Section to a problem in astronomy.

3. The FIRST Survey

The FIRST (Faint Images of the Radio Sky at Twenty-cm) survey [BWH95] is a project that was started in 1993 with the goal of producing the radio equivalent of the Palomar Observatory Sky Survey. Using the Very Large Array (VLA) at the National Radio Astronomy Observatory (NRAO), FIRST is scheduled to cover more than 10,000 square degrees of the northern and southern galactic caps, to a flux density limit of 1.0 mJy (milli-Jansky). At present, with the data from the 1993 through 1999 observations, FIRST has covered about 8,000 square degrees, producing more than 32,000 two-million pixel images. At a threshold of 1mJy, there are approximately 90 radio-emitting galaxies, or radio sources, in a typical square degree. The results we present in this chapter are based on the 1998 version of the catalog, which includes data from 1993 through 1997.

Radio sources exhibit a wide range of morphological types that provide clues to the source class, emission mechanism, and properties of the surrounding medium. Of particular interest are sources with a bent-double morphology as they indicate the presence of clusters of galaxies, a key project within the FIRST survey. FIRST scientists currently use a manual approach to detect bent-double galaxies. They first look at the image of a radio source to see if it could be labeled as a bent-double. If two out of three astronomers agree that the galaxy is a bent-double, then additional observations are carried out in order to study the bent-double in more detail.

This visual inspection of the radio images, besides being very subjective, is also becoming increasingly infeasible as the survey grows in size. Our goal is to bring automation to this process of classifying galaxies by means of techniques from data mining.

Figure 2 includes several examples of radio sources from the FIRST survey. Radio galaxies can have rather complex shapes, as shown in examples (g) through (l). While some bent-doubles are relatively simple in shape (examples (b) and (c)), others, such as the ones in Figure 3 can be rather complex. The task of automating the detection of bent-doubles can be quite complex as seen from the similarity between the bent-double in example (b) and the non-bent-double in example (d).

The data from FIRST, both raw and postprocessed, are readily available on the FIRST website [FIR]. A user friendly interface enables easy access to radio sources at a given RA (Right Ascension, analogous to longitude) and Dec (declination, analogous to latitude) position in the sky.

There are two forms of data available for use — image maps and a catalog. In Figure 3, we show an image map containing examples of two bent-doubles. These large image maps are mostly "empty", that is, composed of background noise. Each map covers an area approximately 0.45 square degrees, with pixels that are 1.8 arc seconds wide. These image maps are obtained as a result of processing the raw data collected by the VLA telescopes.

In addition to the image maps, the FIRST survey also provides a source catalog [WBHG97]. This catalog is obtained by processing an image map by fitting two-dimensional elliptic Gaussians to each radio source. For example, the lower bent-double in Figure 3 is approximated by more than seven Gaussians while the upper one is approximated by three Gaussians. There is an upper limit to the number of Gaussians that are used to fit each radio source. As a result, highly complex sources are not approximated well using just the information in the catalog. Each entry in the catalog corresponds to the information on a single Gaussian. This includes, among other things, the RA and Dec for the center of the Gaussian, the major and minor axes, the peak flux, and the position angle of the major axis (degrees counterclockwise from North). Each of the three entries in the catalog corresponds to one of the three "blobs" in the image. Note that we differentiate between catalog entries and radio sources, with a radio source being composed of one or more catalog entries.

4. Identification of Bent-Doubles

As illustrated in Figure 3, we have data at two extremes. On one hand, we have 200 Gigabytes of image maps, which are mainly noise, with very few "interesting" pixels corresponding to the radio sources. On the other hand, we have the 78 Megabyte catalog information, where each entry contains information on only a part of a radio source. The first task therefore is to identify what constitutes a radio source, using either the image maps, or the catalog, or both. Once we have done this, we need to first identify the potentially relevant features for each galaxy, extract them, and then use them in the identification of the bent-doubles.

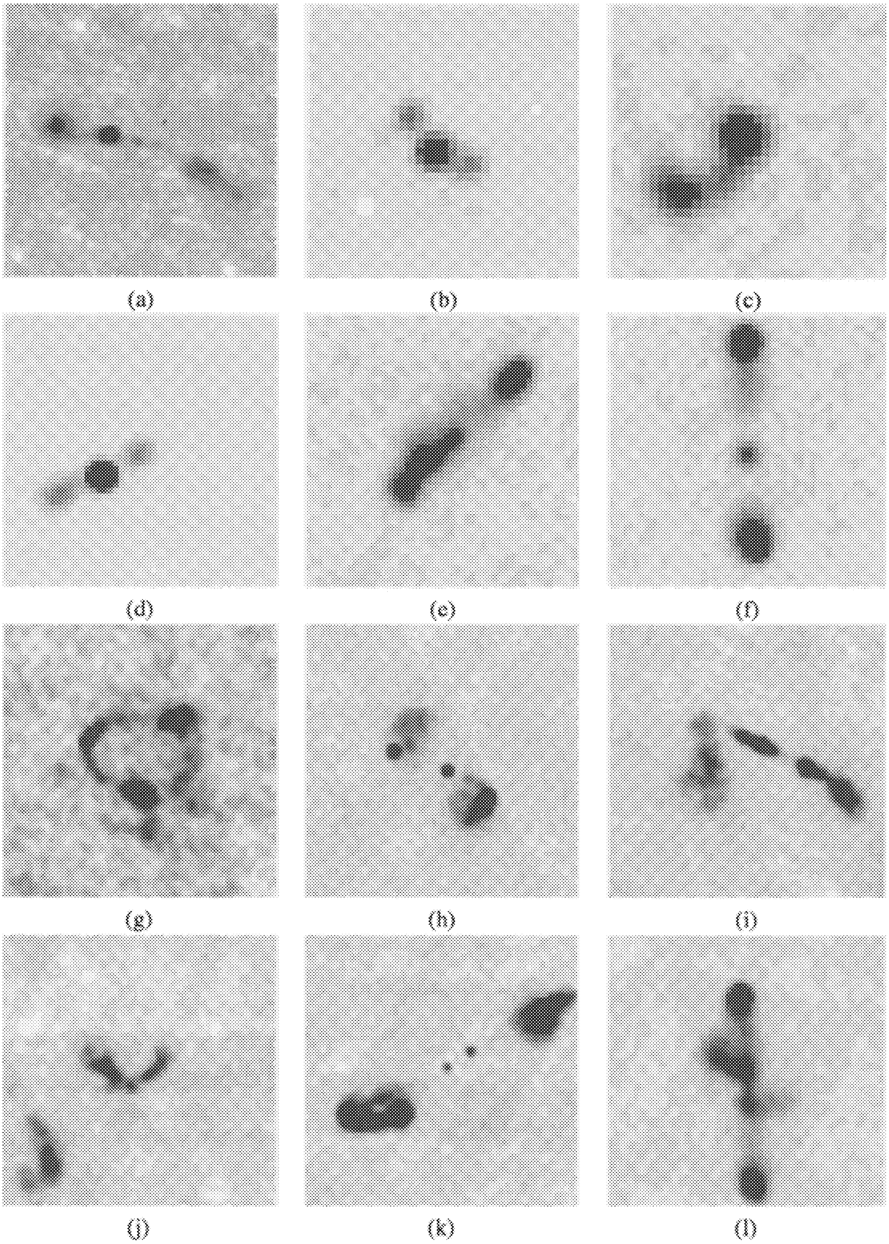

(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)

Figure 2. Example radio sources from FIRST (a)-(c) Bent-doubles. (d)-(f) Non-bent doubles (g)-(l) Complex Sources

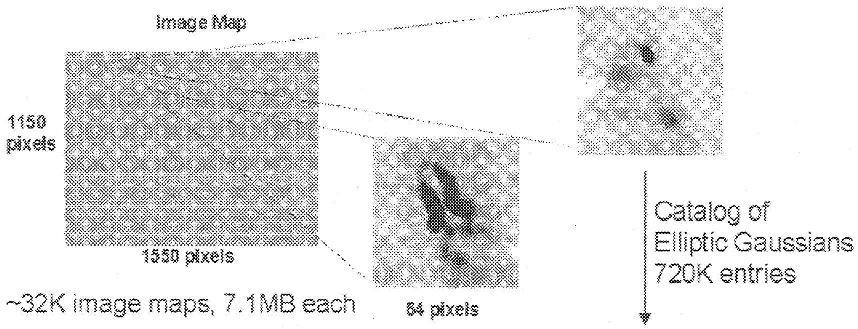


Image Map
1150 pixels
1550 pixels
~32K image maps, 7.1MB each
64 pixels
Catalog of Elliptic Gaussians 720K entries

Catalog entry
Radio Galaxy

RA	DEC	Peak Flux (mJy/bm)	Major Axis (arcsec)	Minor Axis (arcsec)	Position Angle (degrees)
00 56 25	-01 15 43	25.38	7.39	2.23	37.9
00 56 26	-01 15 57	5.50	18.30	14.29	94.2
00 56 24	-01 16 31	6.44	19.34	10.19	39.8

Figure 3. FIRST Data: Images Maps and Catalog Entries.

In our work, we decided that initially, we would identify the radio sources and extract the features using only the catalog. This choice was prompted by several factors:

- The astronomers believed that the catalog was a good approximation to all but the most complex of radio sources.
- It was easier for us to work with the catalog as it was smaller.
- Processing the relatively large image maps for extracting relevant features for the bent-double problem was expected to be difficult and time consuming due to lack of parallel image processing software.
- The FIRST astronomers indicated that several of the features they thought were important in identifying bent-doubles were easily calculated from the catalog.

In Section 5, we comment on the effects of our decision to use only the information in the catalog.

Since we decided to work with the catalog, our first task in classifying the bent-doubles was to group the catalog entries, i.e. the elliptic Gaussians, into radio sources. Our algorithm starts with an entry in the catalog, searches for other entries within a region of interest of 0.96 arc minutes, restarts the search from each newly found entry, and repeats until no more new catalog entries are found within the region of interest. All catalog entries found in this search are collected to form a radio source. Next, the algorithm repeats the entire grouping procedure starting from the next available catalog entry, excluding any entries that are part of already existing radio sources.

In grouping the entries, once a new entry was found within the region of interest, the search could continue from either the center of the new entry, or the center of mass of the entries that made up the source. Our experience indicated that the choice of the starting point had little effect on the resulting grouping.

Note that it is not very difficult to find cases where the catalog entries from one radio source are within 0.96 arc minutes of the catalog entries of a different radio source. For example, Figure 4 with the image centered at RA = $10^h50^m08.5^s$ and Dec = $+30°40'15''$ (J2000 coordinates), shows two radio sources, a bent-double in the lower left corner, and a ring-like structure, which is also a bent-double in the upper right corner. While these radio sources may be far from each other in three dimensions, in a two-dimensional projection, they appear close together. Such examples illustrate one of the many reasons why the task of automated detection of bent-doubles is a rather hard problem. It also shows the ease with which humans can visually identify the two objects as being separate, a task that is difficult to automate.

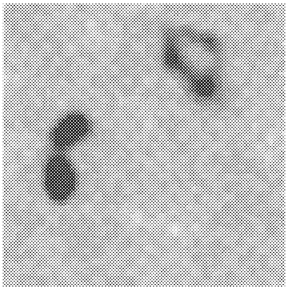


Figure 4. An example image from FIRST, illustrating two galaxies close together

After grouping the catalog entries into complex radio sources, we separated the data depending on the number of catalog entries that make up the sources. There is a data set each for the 1-entry sources, the 2-entry sources, the 3-entry sources, and the 3-plus-entry sources. This separation by the number of catalog entries was done for several reasons. First, we knew that, using features from only the catalog, there were unlikely to be any "bent-doubles" in the single-catalog-entry sources. This was because a single elliptic Gaussian could not be "bent". Further, there are relatively few 3-plus-entry sources, all of which are "interesting" to the astronomers, regardless of whether they are bent-doubles or not. So, we simply flag them and report them to the scientists. This approach also helped us to address the case where there are two radio sources close to each other, with each composed of at least two catalog entries. However, it did not address the case where two disconnected sources, close to each other, were approximated by two or three Gaussians.

Having removed the 1-entry and the 3-plus-entry radio sources from consideration, we further split the sources into two- and three-entry sources. This was done as the number of features extracted depends on the number of catalog entries, and we wanted a feature vector with a uniform length. However, this also meant that the size of the training set for the detection of bent-doubles was now divided into smaller training sets.

For the 1998 catalog, the number of radio sources as a function of the number of catalog entries they are composed of, is as follows:

# Catalog entries	# Radio sources
1	311785
2	40134
3	9235
4+	4765

Once the radio sources (including the training set) were separated based on the number of catalog entries in the galaxy, we derived the features listed in Section 4.1 for the two and three entry sources. Next, using the appropriate training set, we created decision trees for the identification of two- and three- entry radio sources. We ran cross-validation experiments to estimate the accuracy of the tree using different subsets of the features.

4.1. Features for Bent-Doubles

This section describes the features we are using to discriminate galaxies with bent-double morphology. Some of the features are directly taken from the FIRST catalog and others are derived from the basic features in the catalog. We also include a few "features", such as the radio source ID and position in the sky, for bookkeeping purposes only.

Our focus is on features that are scale, rotation and translation invariant, as the bent-double pattern we are looking for, has these properties. We are also interested in features that are robust, that is, not sensitive to small changes in the data [Whi99]. Of course, it goes without saying that the features we select must be relevant to the problem.

We identified the features for the bent-double problem through extensive conversations with FIRST astronomers. As we asked them to justify their decision in identifying a radio source as a bent-double, it became apparent that greater focus was placed on spatial features such as distances and angles. Frequently, the astronomers would characterize a bent-double as a radio-emitting "core" with one or more additional components at various angles, which were usually side-wakes left by the core as it moved relative to the Earth.

We next list some of the key features we calculated based on our collaboration with the FIRST astronomers. A full list of features is described in [FCPKT00]. We broadly categorize the features based on the number of catalog entries that are used in their calculation.

- Features for the radio source

  This includes features that pertain to the entire radio source, such as the number of catalog entries, or bookkeeping features such as the radio source ID (for tracking purposes), or the hemisphere for the radio source (nothern or southern).

- Features for each catalog entry

  This includes features pertaining to a single catalog entry, such as its peak flux, total area of the elliptic Gaussian, the ellipticity of the Gaussian, the major and minor axes, etc. We also include the position angle, which is the angle (in degrees) of the major axis, measured counterclockwise from North. In Figure 5, the angles are indicated by an arrow — for entry B it is about 45^o in the left example, and about $(180 - 45)^o$ in the right example. The angle is 0 for entry A in both examples. Note that the position angle is not a robust feature as it is very sensitive to minor changes in the image.

- Features for each pair of catalog entries

  This category includes two types of features. In the case of two-catalog-entry radio sources, these are features representing the radio source, such as the total area (the sum of the areas of the two Gaussians) and the peak flux (the larger of the two fluxes). In addition, there are features that are obtained by considering catalog entries two at a time. These include relative distance between two entries, and the pair-wise geometric angle, which is the angle formed by the position angles of the two major axes, as calculated geometrically – angle AMB in both examples of Figure 5. We also include the absolute difference in the position angles of the two entries, which is about $|0 - 45|^o = 45^o$ in the left example, and about $|0 - 135|^o = 135^o$ in the right example of Figure 5. Note that this feature is not robust to small changes in the image.

- Features for each triple of catalog entries

  In the case of three-entry sources, some features, such as the total area (sum of the areas of the Gaussians), represent the entire radio source. Others include various measurements of angles, such as the angle sub-

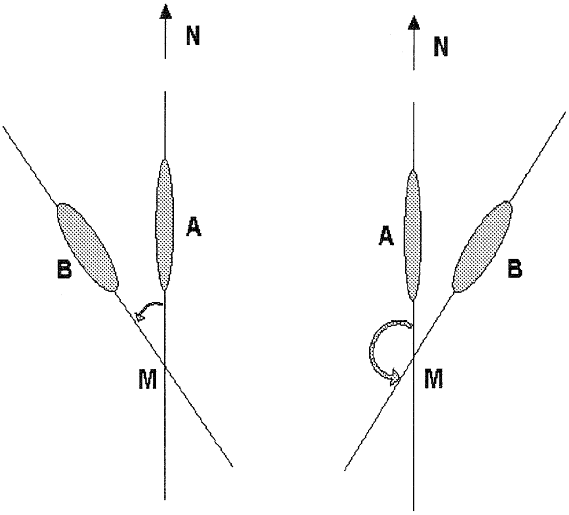


Figure 5. Two examples of 2-entry fitted radio sources.

three Gaussians [LBR+99], and the angle subtended by the two sides of the triangle that are closest to each other in legnth.

In the case of two-entry sources, the features were listed in decreasing order of integrated flux for each Gaussian. The features include the ones for the radio source, the ones for each catalog entry and the ones for the pair of catalog entries. In the case of 3-entry radio sources, we include all four categories of features described earlier. We also explored three ways of ordering the three entries. These were based on first identifying one of the catalog entries as the "core" of the radio source. The options we considered include:

I. Choose the entry with the largest integrated flux as the core. Order the entries as: A (maximum integrated flux), B (second largest integrated flux), C (smallest integrated flux). Note that this is a some-what ad-hoc ordering, with no real astronomical basis behind it.

II. Choose the core as the entry opposite the largest side of the triangle formed by the centers of the three ellipses. Order the entries as: A (opposite largest side), B (opposite second largest side), C (opposite smallest side).

III. Choose the core to be the entry opposite the side that is most "unlike" the other two sides. Order the entries as: A (the center such that the two sides of the triangle that meet at that center are closest in length), B (the center such that the two sides of the triangle that meet at that center are second closest in length), C (the center such that the two sides of the triangle that meet at that center are farthest in length).

For the 3-entry sources, we repeated the feature extraction step separately for each of the three ordering methods, and ran the decision tree algorithm on the three different sets of features.

5. Results Using Decision Trees

In this Section, we present some of our early results using decision tree classifiers [BFOS84, Qui93] and the features from the catalog described in Section 4.1. We first make the following observations:

- Our training set is relatively small, with 118 examples for two-catalog entry sources and 195 examples for the three-catalog entry sources. As the bent- and non-bent-doubles have to be manually labeled by FIRST scientists, putting together an adequate training set is a non-trivial task. We plan to enhance our small training set by using feedback from the astronomers on the results of the preliminary decision trees.
- Scientists are usually subjective in their labeling of galaxies as bent- or non-bent-doubles. There is often disagreement among the astronomers in the hard-to-classify cases. There is also no ground truth we can use to verify our results. This implies that the training set itself is not very accurate, and there is a limit to the accuracy we could obtain through the use of semi-automated techniques.
- We are currently using features from only the catalog. We would therefore expect that if the "bentness" of a radio source was adequately captured by the catalog, we would do well in identifying a bent-double.

In our experimentation, we expect that some of the features will not be important in finding bent-doubles. For example, the position in the sky, that is, the (RA, Dec) coordinates, should not influence the results, at least as long as bent-doubles are approximately randomly distributed over the celestial sphere. However, our initial experiments with decision trees indicated that the coordinates were influential. On further investigation, we realized that when the astronomers provided us examples of non-bent-doubles to use in our training set, they had focused on a small section of the sky, thus making the coordinates influential. In this case, the decision tree was "right", but there was a problem in the features we used in training. While we expected the decision tree to focus on the features which are discriminating, this experiment illustrated the important role played by domain knowledge in the selection of features. As a result, in the remaining sections, we exclude all the bookkeeping "features" and the position coordinates from the analyses.

Method	*Tree Size*	*Errors*
Method I.	7.3 (0.1)	10.9 (0.6)%
Method II.	7.3 (0.1)	11.8 (0.6)%
Method III.	5.9 (0.0)	9.6 (0.2)%

Table 1. Results of ten 10-fold cross-validation experiments for the three different ordering methods.

5.1. Results for 3-Entry Sources

For the three catalog entry sources, the training set consists of 195 labeled examples, with 167 bent-doubles and 28 non-bent-doubles. Using the features and methods listed in Section 4.1, we repeated 10-fold cross-validation experiments 10 times for each of the three ordering methods (100 trees per method in total). In each experiment, the training set is first randomly divided into ten parts, and the decision tree grown based on nine parts at-a-time, is validated on the remaining one part. The results are given in Table 1. The tree sizes and errors on each line are the means of the hundred trees, with the standard error in parenthesis. The errors combine both misclassifications: bents classified as non-bents, and non-bents classified as bents. The astronomers tolerate higher rates of the latter errors, but would like to minimize the mistakes of the former type.

As expected, ordering method III gives the most accurate results. Bent-doubles generally exhibit a symmetry around the core, so this method makes the most sense out of the three considered.

We expected method II to be the next best performer, but, to our surprise, method I gave better results. Our astronomer collaborators indicate that there is no relationship between the flux magnitude and the location of the core, so we are unable to explain this result at present. Selecting the core according to the largest angle, i.e. method II, gave the worst results. We thought it would be superior to method I, as there is greater connection between the geometry of the source and bentness, than there is between the flux and bentness. There are many bent-doubles with the largest angle at the core, so we expected method II to be closer to method III. The latter picks up the two different types of symmetries (core is the largest, or core is the smallest angle), while the former only considers one of the symmetries (core is the largest angle). We are exploring these issues in greater detail in order to fully interpret the results. Note, however, that while the errors are slightly different for the three ordering schemes, they are on the same order. Also, these results must be interpreted keeping in mind that the size of the training set is relatively small at present.

A typical tree constructed with ordering method III is given below.

```
Decision tree:

rs3_core_angle > 170.4:
:...cec_ellipticity <= 2.116: 1 (13.0)
:   cec_ellipticity > 2.116: 5 (2.0)
rs3_core_angle <= 170.4:
:...pairac_rel_dist <= 9.423: 5 (143.0)
    pairac_rel_dist > 9.423:
    :...pairab_angle_geom <= 58.6: 5 (4.0/1.0)
        pairab_angle_geom > 58.6:
        :...cec_rms <= 0.137: 5 (5.0/2.0)
            cec_rms > 0.137: 1 (9.0)

Evaluation on training data (176 cases):

            Decision Tree
          ----------------
          Size      Errors
             6    3( 1.7%)   <<

           (a)   (b)    <-classified as
          ----  ----
            22     3    (a): 1 (non-bent)
                 151    (b): 5 (bent)

Evaluation on test data (19 cases):

            Decision Tree
          ----------------
          Size      Errors
             6    2(10.5%)   <<

           (a)   (b)    <-classified as
          ----  ----
             1     2    (a): 1 (non-bent)
                  16    (b): 5 (bent)
```

The decision tree output lists the feature selected at each node, as well as the value it is compared against. The number after the colon indicates that the node in question is a leaf node, and the number is the class assigned to the leaf (5 denotes a bent-double, while 1 denotes a non-bent double). At each leaf node, the numbers (a/b) indicate the (total number of samples/samples of the class not assigned to leaf node).

	Tree Size	*Errors*
Single	11.2 (0.1)	19.7 (0.5)%
Double	8.7 (0.2)	17.4 (0.4)%
Single+double	10.7 (0.2)	19.2 (0.5)%
Triple	6.7 (0.1)	10.7 (0.3)%
Single+triple	6.4 (0.0)	8.5 (0.4)%
Double+triple	7.1 (0.1)	11.6 (0.5)%
Single+double+triple	5.9 (0.0)	9.6 (0.2)%

Table 2. Average of ten 10-fold cross-validation experiments for each of the seven 3-entry feature combinations.

For the 3-entry cases, the decision trees based on ordering III tend to pick combinations of angles and relative distances as the most important features to discriminate bent-doubles. Other features deemed important include measures of ellipticity and symmetry — features that are all scale, rotation, and translation invariant. The angles are usually either the core angle, or pairwise angles calculated geometrically — angles that are robust to small changes in the data. The very reason we included these pairwise geometrical angles (Section 4.1), was to avoid using the more sensitive feature that measured the absolute difference in the position angles. The trees generally ignore features related to the fluxes and the areas. Overall, the trees make sense, and they pick the features that we expected to be closely related to bent-doubleness.

The trees based on the other two ordering schemes were not as consistent as the ones corresponding to the ordering method III. The discriminating features selected occasionally included flux and area measurements, and major axes lengths, in combination with distance and angle values. A few trees that we examined selected actual, rather than relative, distance measurements. The actual distances, and other features such as flux, area, and major axis, are poor discriminating features, as they are not strictly scale invariant. The brightness, and the size of an entry should not be related to bent-doubleness. Our initial experiments thus indicate that, given the current training set, the ordering methods I and II are inferior to ordering method III in classifying bent-doubles. They have relatively high accuracy, but, on closer examination, they base the classification on features that do not make sense from the domain science point of view, and that keep changing from tree to tree, depending on the training and validation sample selected.

To reduce the number of features, we next repeated the decision-tree building steps, using combinations of the single, double, and triple features. The results for 10 different 10-fold cross-validations for each of the seven combinations, based on the ordering method III are presented in Table 2.

The table reinforces our expectation that the most important features are the triple ones. Using only the triple features, the misclassification rate is 10.7%(0.3%), a small increase from the 9.6%(0.2%) achieved when using all the features. The single and/or double features by themselves lead to close to 20% errors. Adding the double features to the triples slightly degrades the results, while adding the singles to the triples improves the results. This behavior may be due to the existence of redundant features; it is currently under investigation.

An early version of our decision tree, when used for classifying unlabeled data, found several new bent-doubles, as expected. For example, Figure 6 shows two examples of new bent-doubles from the region the astronomers had not looked at manually (left panels). What is interesting is that the data mining process also found a bent-double that the astronomers had missed (right panel) during the visual inspection that generated the training set. This illustrates one of the many benefits of data mining techniques in the semi-automated exploration of massive data sets.

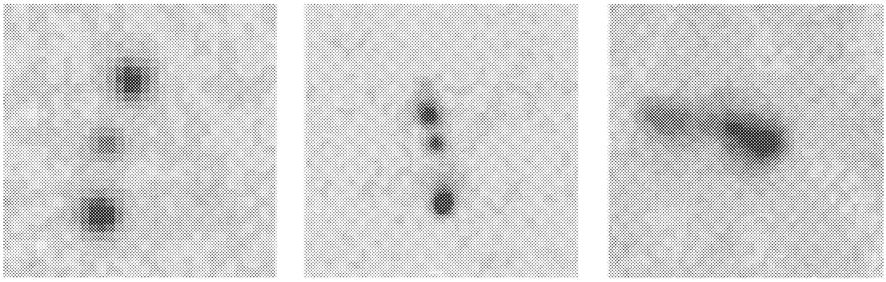


Figure 6. Examples of two new bent-doubles and one bent-double overlooked in a manual search

5.2. Results for Two-Entry Sources

Our initial experiments with two-entry sources found that the observations made in Section 5 play an important role. Using only catalog-based features with the limited training set, the decision trees created were erratic. In cross validation experiments, we found that the tree strongly depended on the subset selected from the full training set. The misclassification errors that resulted were also relatively high, on the order of 20%.

We suspect there are several reasons for this, including the small training set (118 examples, with 72 bent-doubles and 46 non-bent-doubles) and possible irrelevant features. In addition, we suspect that in the case of two-entry sources, the catalog may not be an adequate representation of all the relevant information required to correctly identify a bent-double. For example, Figure 7

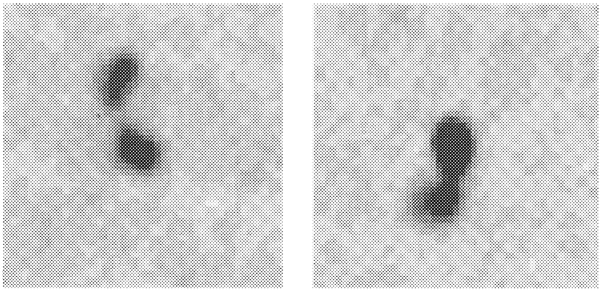


Figure 7. Examples of 2-entry galaxies - a bent-double and a non-bent-double.

includes two entries from our training set, one a bent-double and the other a non-bent-double. Some astronomers have indicated that the faint bridge in the left example (which is not captured in the catalog) makes it a bent-double. In addition, these examples fall in the hard-to-classify case, making our training set not very accurate.

We therefore defer a detailed analysis of the 2-entry sources until later, when we can refine the features, add image-based features, and increase the training sample.

6. Conclusions and Future work

In this chapter, we described how data mining techniques can help astronomers detect radio galaxies with a bent-double morphology in a semi-automated manner. While we can by no means claim to understand and appreciate all the challenges that lie in mining scientific and engineering datasets, our experiences with this problem has led to the following observations:

- Scientific data is frequently available in the form of "raw" data, that is, data from which features have not been extracted. In our experience, the identification and extraction of relevant features in a robust manner is non-trivial and results in one of the more time consuming steps in data mining. In our specific example of the detection of bent-doubles, the existence of the catalog was of immense help in getting a head-start on the problem.

- Easy access to the data, as well as the existence of good public-domain software to read, write, and display the data are essential in order for data miners to work with scientific data sets. In the case of the FIRST data, the availability of the data on the web, as well as tools to read/write the astronomy data stored in the FITS format, were very helpful.

- In many, if not most, data mining endeavors, it takes some time for the data miners to understand the problem domain, the problem, and the data itself. As many data miners are typically not familiar with the intricacies of the issues involved in scientific data, as opposed to say commercial data, this time can be considerable. In our experience, we found it took six months to understand the bent-double problem, the data formats, the processing that had already been done to the data, etc.
- In scientific data, for classification problems, there can be a dearth of labeled examples. This is because such examples must be manually identified by the scientists. This is in contrast with commercial data, where there is the possibility of labeled examples being generated historically, for example, in customer churn problems. Further, labels for scientific data are typically subjective, and we found it common to have astronomers disagree on whether a galaxy should be classified as a bent-double or a non-bent-double. This was especially true in the cases which were difficult to classify. Given the lack of ground truth in this problem, generating a good training set has been difficult.
- Since FIRST is a survey in progress, we found that our bookkeeping had to keep up with changes made in different releases of the data. For example, the 1999 version of the data merged the information from the northern and southern hemisphere, information that had previously been separate. Also, in the 2000 version, we found that certain galaxies no longer appeared in the catalog. Conversations with astronomers indicated that this was a normal occurrence as a result of the processing of the data that was collected by the telescopes. For galaxies that were at the very edge of the survey in one year, additional data collected in the following year would make the pixels corresponding to the galaxies fall below the detection threshold. This meant that we had to be careful in our use of the ID tags for galaxies as we move to newer versions of the survey.

Our initial experiences with the bent double problem appear promising, though much remains to be done. In the near term, we plan on increasing the size of the training set, revising the catalog-based features, and adding image-based features. Revising the catalog-based features has been an ongoing process. For the three-entry sources, our average misclassification rate of about 10% is half the rate we initially obtained during the first iteration of the data mining process. New features emerge as we discuss our findings with our astronomer collaborators. We are also improving the features derived from the catalog by removing possible redundancies among the various angle and distance measurements by combining them into fewer, more relevant features. We also plan on using other pattern recognition techniques such as neural networks to see how they perform on the bent-double problem.

Acknowledgments

We gratefully acknowledge our FIRST collaborators Robert Becker, Michael Gregg, David Helfand, Sally Laurent-Muehleisen, and Richard White for their technical interest and support of this work. We would also like to thank Charles Musick, Deanne Proctor, and Ari Buchalter for useful discussions and/or computational help.

UCRL-JC-140418. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* CRC Press, 1984.

[BWH95] R. H. Becker, R.L. White, and D.J. Helfand. The FIRST survey: Faint images of the radio sky at twenty-cm. *Astrophysical Journal*, 450:559, 1995.

[FCPKT00] I. K. Fodor, E. Cantú-Paz, C. Kamath, and N. Tang. Finding bent-double radio galaxies: A case study in data mining. In *Interface : Computer Science and Statistics*, volume 33, April 2000.

[FIR] FIRST: Faint images of the radio sky at twenty centimeters. http://sundog.stsci.edu/.

[KBFT00] C. Kamath, C. Baldwin, I. Fodor, and N. Tang. On the design and implementation of a parallel, object-oriented, image processing toolkit. In *Proceedings International Symposium on Optical Science and Technology, SPIE Annual Meeting, San Diego*, July 2000.

[KCP00] C. Kamath and E. Cantú-Paz. On the design of a parallel object-oriented data mining toolkit. In *Workshop on Distributed and Parallel Knowledge Discovery at the Knowledge Discovery and Data Mining Conference Boston*, August 2000.

[KM00] C. Kamath and R. Musick. Scalable data mining through fine-grained parallelism: The present and the future. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 29–77. AAAI Press/The MIT Press, 2000.

[LBR$^+$99] J. Lehar, A. Buchalter, R.McMahon, C.Kochanek, D.Helfand, R.Becker, and T.Muxlow. The FIRST efficient gravitational lens survey. 1999. submitted to "Gravitational Lensing: Recent progress and Future Goals, eds: T.Brainerd and C.Kochanek, ASP Conf Series See also http://xxx.lanl.gov/abs/astro-ph/9908353.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[Sap] Sapphire: Large-scale data mining and pattern recognition. http://www.llnl.gov/casc/sapphire.

[WBHG97] R. L. White, R.H. Becker, D.J. Helfand, and M.D. Gregg. A catalog of 1.4 GHz radio sources from the FIRST survey. *Astrophysical Journal*, 475:479, 1997.

[Whi99] R. L. White, 1999. Private Communication.

Chapter 7

A DATASPACE INFRASTRUCTURE FOR ASTRONOMICAL DATA

Robert Grossman, Emory Creel, Marco Mazzucco, and Roy Williams

Abstract This article describes an internet infrastructure for working with data called DataSpace. A distributed DataSpace application containing data from the 2MASS and DPOSS astronomical data sets is also described. DataSpace is designed so that client applications supporting the remote analysis and distributed mining of data are easy to build.

Keywords: data mining, distributed data mining, astronomy, virtual observatory, protocols, DSTP

1. Introduction

The web today provides an infrastructure which is better suited for working with multimedia documents than for working with data. In this article, we describe an internet infrastructure called DataSpace, which is designed to support the publishing, analysis, and mining of remote and distributed data. We also show how DataSpace can be used to work with astronomical data, and, in particular, to create virtual observatories.

The core technology is a network based transfer protocol called the data space transfer protocol (DSTP) [SBS00], a platform independent way to share data over a network. It is based on a data model which views data as a collection of distributed columns which can be linked for analysis through "global keys" called universal correlation keys (UCK's). In this way, DataSpace provides a means for remote data analysis and distributed data mining which is independent of the type of data storage used, whether it be files, databases, or distributed data warehouses.

The DSTP protocol enables an application in one location to locate, access, and analyze data from several other locations. It also reduces the dependency on the data file, because it correlates data based on UCK's in different data sets. The DSTP protocol provides the infrastructure for a

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 115–123.
© 2001 *Kluwer Academic Publishers.*

Description	Document Web	Data Web
retrieve	retrieve a remote document	analyze & mine remote data
search & mine	search engines using keyword search	data mining engines analyzing distributed data
language	HTML, XML	XML, PMML [PMM], SOAP
protocol	HTTP	DSTP
server	web server	data server
hardware	workstation	workstation clusters
network	10K document can be retrieved < 1 sec with commodity internet	1 MB data set retrieved < 1 sec with OC-3 network

Table 1. Next generation networks with high bandwidth, quality of service and security will enable the emergence of a data web, broadly analogous to today's document web.

global *data space* in the same way as the http provides the infrastructure for a global *document space.*

Section 2 describes related work and provides some background material. Section 3 contains a quick introduction to DataSpace. Section 4 describe the virtual observatory we created with DataSpace. Section 5 shows the DSTP protocol underlying a typical query to the virtual observation. Section 6 contains the conclusion and summary.

2. Related Work

Today there are several common formats for scientific and engineering data available on the web:

> *Flat Files.* The most common format is probably files, either simple ASCII files with data arranged in rows and columns, self describing file formats such as HDF [hdf], or binary formats, accompanied by programs for extracing and working with the data. Often the relevant data file is selected using a cgi program which allows the user to query on the metadata.
>
> *html Tables.* If the data set is small enough, HTML tables can be used so that the data is directly available for viewing.
>
> *Databases.* Another common format is to put the data into a database and to provide access to the database using a form to produce an sql query, which is passed to the database via

an API (application program interface) such as ODBC (open database connectivity) [San98] or JDBC (java database connectivity) [Ree00]. If the number of records returned is small, the records can be directly displayed in HTML; if not, the records can be output to a file.

XML. Some data is beginning to be stored in XML (extensible markup language), especially if the data sets are small.

Access to data is usually through a cgi program. If the data is in a database, access may involve ODBC or JDBC. Sometimes agents may be used. Access may also involve SOAP (simple object access protocol) [SOA] and HTTP.

Unless specialized programs are available for acessing file based data, file based data usually requires that the entire data be transferred before analysis can begin. Although databases are efficient for writing, selecting and reading records, they are not designed to provide a mechanism for the efficient statistical analysis of data. Data in HTML tables can easily be displayed, but not otherwise manipulated. Finally, XML provides a very flexible import and export format, but does not directly support the analysis of data. Also, it is very verbose which is not efficient for large data sets. DataSpace is designed to provide an alternative mechanism for storing, accessing, and analyzing data, which is complementary to the formats and access methods just described.

3. Data Space

In this section, we give a quick introduction to some of the key concepts underlying DataSpace.

Data Sites. A data site is simply a web site with a DSTP server listening on port 5040 for requests from DSTP clients.

Catalogs and Metadata. Adding data to a site is easy.

1 First, create a text file, say dposs.dat containing rows and columns of data.

2 Second, add the name of the text file to one of the catalog files at the site, say catalog.dstp, which contains the names of all the data files.

3 Third, add a file dposs.ds containing simple metadata about the data file, such as the names of the attribute, their data types, etc.

Columns of Data. Although data is physically loaded into a data site using files, DataSpace logically organizes the data into columns and provides efficient physical access to one or more columns of data.

Universal Correlation Keys. Each column of data is associated with one or more universal correlation keys or (UCK's). Each UCK is associated with a globally unique identifier. Two distributed columns of data with the same UCK may be meaningfully compared. An example of a UCK the number of seconds since January 1, 1970. Another example is right ascension and declination measured in degress.

4. A DataSpace Based Virtual Observatory

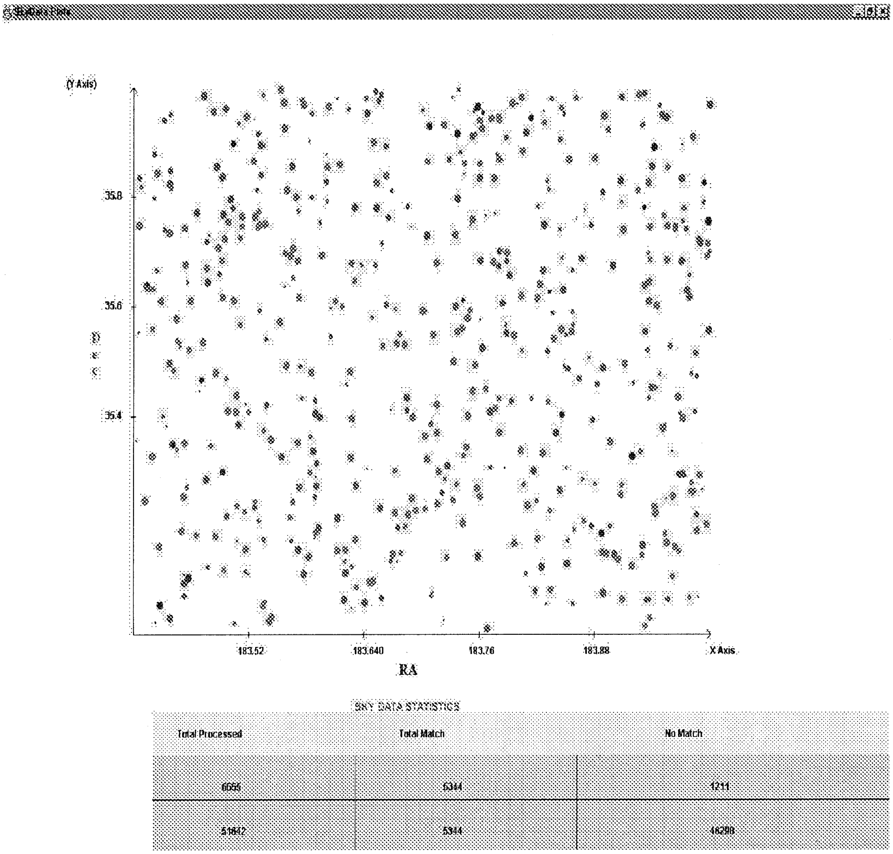
Using the dataspace infrastructure, we developed an application which simultaneously works with two geographically distributed astronomical source catalogs: 2MASS (Two Micron All Sky Survey) survey [2MA] data from CalTech and DPOSS (Digital Palomar Observatory Sky Survey) survey [DPO] data. The 2MASS data are in the optical wavelengths (0.4 - 0.7 micron), while the DPOSS data are are in the infrared (1.2 - 2.2 micron) range.

Our goal was to create a virtual observatory supporting the statistical analysis of many millions of stars and galaxies with data coming from both surveys. In order to support this type of analysis, we must identify the same star in both surveys. It is a simple matter: if two star images are in the same position, they must be the same star. However, each star has a finite angular size in the sky, so there is a slight "fuzziness" in position, and this is exacerbated for galaxies. A typical query is of the form: "Find all pairs, one from the DPOSS catalog and one from 2MASS, whose angular separation is less than a given tolerance".

In the DSTP application we built, this query is visualized by coloring the data as follows:

- red (appears only in 2MASS), or
- blue(appears only in DPOSS), or
- magenta (successful join: appears in both surveys).

At CalTech, we populated one DSTP server with 2MASS data. For the studies described, we populated another DSTP server with DPOSS data. This was easier than putting a second DSTP server at the DPOSS home site. The DPOSS DSTP server contains a list of stars within a given region of the sky, with magnitudes in three different colors. The 2MASS



SKY DATA STATISTICS

Total Processed	Total Match	No Match
6555	5344	1211
51642	5344	46298

Figure 1. DSTP client

DSTP servers contains a list of stars in a given region, accompanied by extensive data about the stars.

The client DSTP application formulates and sends requests to a compute server, which interprets the client's request, gets the relevant data from the DPOSS and 2MASS servers, performs the fuzzy join, and sends the resulting stars and galaxies back to the client.

An architecture like this, which uses separate servers for data and computational tasks, can scale to large numbers of distributed services which may be communicating very large amounts of data at very high bandwidth.

This virtual observatory we developed was introduced at Supercomputing 2000 in Dallas in early November, 2000. Figure 1.1 is a sample output of our client. The position of the light sources in both data sets were given in polar coordinates, right ascension and declination. For this experiment a tolerance of 2 seconds was chosen. This tolerance was suggested to us physicists at California Institue of Technology familiar with the two data sets. The data was raw data, with out any attempt to eliminate noise. Also, in order to achieve a reasonable graph, we restricted the data in both catalogs to the portion of the sky in the right ascension range of 183 to 270 degrees and declination in the range of 17 to 47 degrees (somewhere over the north pole).

As can be seen, a match rate of about 82 percent was observed between these two data sets. Something that can not be observed from a screen shot is the continuous change of both the graph and the statistics below the graph during the runtime of this application. This is one of the many benefits of the DSTP protocol over a simple file transfer, its ability to stream data and allow a client to process the data in real time.

5. A Sample DSTP Session

The Data Space Transfer Protocol (DSTP) is a protocol for retrieving data from remote nodes in the internet. It is broadly based upon the Network News Transfer Protocol (NNTP) [Lap86], a protocol for retrieval of news articles. The DSTP server uses a stream connection and NNTP-like commands and responses. It is designed to accept connections from hosts and to provide a simple interface to the data columns on the server. A DSTP server functions as an interface between DSTP applications and remote data.

Rather than describe the protocol and commands formally, this section contains an annotated session between a DSTP Server named S and a DSTP client named C (not their real names). First we see the client contacting the server on tcp port 5040.

```
C: Trying 131.193.181.121...
C: Connected to ncdm121.lac.uic.edu.
C: Escape character is '^]'.
S: 200 ncdm121.lac.uic.edu DSTP server v0.90 ready
S: .
```

The client can now request which universal correlation keys (UCK's) the server is prepared to serve. The right ascension is assigned the UCK 10 and the declination is assigned the UCK 11, which are the polar coordinates for the sky data.

```
C: list uck
```

```
S: 214 List of UCKs follows
S: <UCK ID="10" NAME="ra" NUMATTR="42"/>
S: <UCK ID="11" NAME="dec" NUMATTR="42"/>
S: <UCK ID="91" NAME="designation" NUMATTR="42"/>
S: <UCK ID="20" NAME="j_m" NUMATTR="42"/>
S: <UCK ID="21" NAME="h_m" NUMATTR="42"/>
S: <UCK ID="22" NAME="k_m" NUMATTR="42"/>
S: <UCK ID="23" NAME="id_opt" NUMATTR="42"/>
S: <UCK ID="30" NAME="b_m_opt" NUMATTR="42"/>
S:  <etc> ...
S:.
```

We assume now, for this example, the client has access to another DSTP server which also can serve the UCK's 10 and 11. So the client requests that the server set these to indexes for correlation.

```
C: set uck 10
S: 211 UCK 10 ra selected
S: .
C: set uck 11
S: 211 UCK 11 dec selected
S: .
```

Next the client can ask what data files the server has which contain these UCK's:

```
C: list uck datafile
S: 234 List of Datafiles follows
S: <DATAFILE NAME="2mass.dat" NUMRECORDS="2912952" DSFILENAME="2mass.ds"
   UCKNAME="ra" UCKID="10" NUMATTR="42" UCKNAME="dec" UCKID="11"
   NUMATTR="42" UCKNAME="designation" UCKID="91" NUMATTR="42"
   UCKNAME="j_m" UCKID="20" NUMATTR="42" UCKNAME="h_m" UCKID="21"
   NUMATTR="42" UCKNAME="k_m" UCKID="22" NUMATTR="42" UCKNAME="id_opt"
   UCKID="23" NUMATTR="42" UCKNAME="b_m_opt" UCKID="30" NUMATTR="42"
   UCKNAME="r_m_opt" UCKID="31" NUMATTR="42" UCKNAME="glon"
   UCKID="13" NUMATTR="42" UCKNAME="glat" UCKID="14" NUMATTR="42"
   UCKNAME="j_h" UCKID="40" NUMATTR="42" UCKNAME="h_k" UCKID="41"
   NUMATTR="42" UCKNAME="j_k" UCKID="42" NUMATTR="42"/>
S: <DATAFILE NAME="2mass_s.dat" NUMRECORDS="536" DSFILENAME="2mass_s.ds"
   UCKNAME="ra" UCKID="10" NUMATTR="5" UCKNAME="dec" UCKID="11"
   NUMATTR="5" UCKNAME="j_m" UCKID="20" NUMATTR="5" UCKNAME="h_m"
   UCKID="21" NUMATTR="5" UCKNAME="k_m" UCKID="22" NUMATTR="5"/>
S: .
```

Then the client selects the desired data file:

```
C: set datafile 2mass.dat
S: 230 Datafile 2mass.dat selected
S: .
```

After the data file is selected, the client can request the corresponding metadata:

```
C: metadata
S: 250 Metadata follows
S: <DATAFILE NAME="2mass.dat" NUMRECORDS="2912952" DSFILENAME="2mass.ds"/>
S: <UCKNAME="ra" UCKID="10" NUMATTR="42" />
S: <UCKNAME="dec" UCKID="11" NUMATTR="42" />
S: <UCKNAME="designation" UCKID="91" NUMATTR="42" />
S: <etc>
S: <ATTRIBUTE-DESCRIPTOR NUMBER="1" NAME="ra" DATA-TYPE="real"
   USE-AS="continuous" UNIT="deg" />
S: <ATTRIBUTE-DESCRIPTOR NUMBER="2" NAME="dec" DATA-TYPE="real"
   USE-AS="continuous" UNIT="deg" />
S: <ATTRIBUTE-DESCRIPTOR NUMBER="3" NAME="err_maj" DATA-TYPE="real"
   USE-AS="continuous" UNIT="arcsec" />
S: <ATTRIBUTE-DESCRIPTOR NUMBER="4" NAME="err_min" DATA-TYPE="real"
   USE-AS="continuous" UNIT="arcsec" />
S: <ATTRIBUTE-DESCRIPTOR NUMBER="5" NAME="err_ang" DATA-TYPE="integer"
   USE-AS="continuous" UNIT="deg" />
S: <etc>
.
```

Finally the client requests the data itself. First the client sets the range for the rows of interest, 10 through 20, in this example. Then the client issues the data command along with a list of attributes which are desired.

```
C: set line 10 20
S: 270 Lines from 10 to 20 selected
S: .
C: data 1 10 34
S: 242 [1, 10, 34] retrieved - data follows
S: 183.400288,27.646980,183.400288,14.797,0.90
S: 183.400334,39.956638,183.400334,10.999,1.84
S: 183.400358,41.809029,183.400358,12.889,0.86
S: <etc>
S: .
```

Finally the client closes the connection with the server:

```
C: quit
S: 205 GoodBye
S: .
C: Connection closed by foreign host.
```

6. Discussion and Summary

In this article, we have described how DataSpace can be used to provide a simple means to analyze remote astronomical data, such as the DPOSS and 2MASS surveys. We have also illustrated how DataSpace provides a simple way to support distributed data mining queries, such as the fuzzy match identifying objects appearing in both catalogs. With DataSpace, virtual astronomical observatories can easily be built and be used to provide access to a wide range of astronomical data. Moreover, with the universal correlation keys used by DataSpace, survey data can also be used by other applications.

7. Acknowledgements

The DSTP based virtual observatory was created by a project team co-led by Robert Grosman and Roy Williams. The team consisted of Shirley Connelly, Emory Creel, Sivakumar Harinath, Marco Mazzucco, Andriy Turinskiy, and Scott Wahlstrom.

References

[2MA] *http://www.ipac.caltech.edu/2mass/.*

[DPO] *http://astro.caltech.edu/ rrg/science/dposs.html.*

[hdf] *http://hdf.ncsa.uiuc.edu.*

[Lap86] Phil Lapsley. *Network News Transfer Protocol.* 1986.

[PMM] *http://www.dmg.org.*

[Ree00] George Reese. *Database Programming with JDBC and Java, 2nd Edition.* O'Reilly, 2000.

[San98] Roger Sanders. *Hans on ODBC 3.5 Developer's Guide.* Osborne McGraw-Hill, 1998.

[SBS00] R. Grossman S. Gutti S. Bailey, E. Creel and H. Sivakumar. A high performance implementation of the data space transfer protocol (dstp). In M. J. Zaki and C.-T. Ho, editors, *Large-Scale Parallel Data Mining*, pages 55–64. Springer-Verlag, 2000.

[SOA] *http://www.w3.org/SOAP/.*

Chapter 8

DATA MINING APPLICATIONS IN BIOINFORMATICS

Naren Ramakrishnan and Ananth Y. Grama

Abstract This chapter describes opportunities for data mining in the emerging arena of bioinformatics applications. We outline the nature of research issues in bioinformatics and the motivating data management and analysis tasks. Descriptions of successful applications are given, along with an outline of the near-future potential and issues affecting the successful application of data mining.

Keywords: Bioinformatics applications, structural genomics, functional genomics, microarray analysis, proteomics, and regulatory networks.

1. Introduction

Rapid advances in scientific computations are being heralded by paradigm shifts in computing infrastructures and application support systems. While new computing platforms — ranging from networked supercomputing clusters and Grids to metasystems — have been explored, new programming primitives for application support have also been formulated. One recent development has been the emergence of data mining as a primary mode of investigation in computational simulation, complementing traditional analysis, visualization, and design tasks. Consequently, support for data-intensive computing in scientific simulations has become increasingly important [CBCR98; MBM$^+$98; MPE98; RB96].

As data mining emerges as a key enabling technology, there have been efforts directed at the consolidation of underlying analysis techniques across various domains. This consolidation hinges on the abstraction of basic analysis tasks, nature of data sources, the computational environment, and the desired result(s). While this effort has achieved some success, for example, in the development of domain independent asso-

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 125–139.
© 2001 *Kluwer Academic Publishers.*

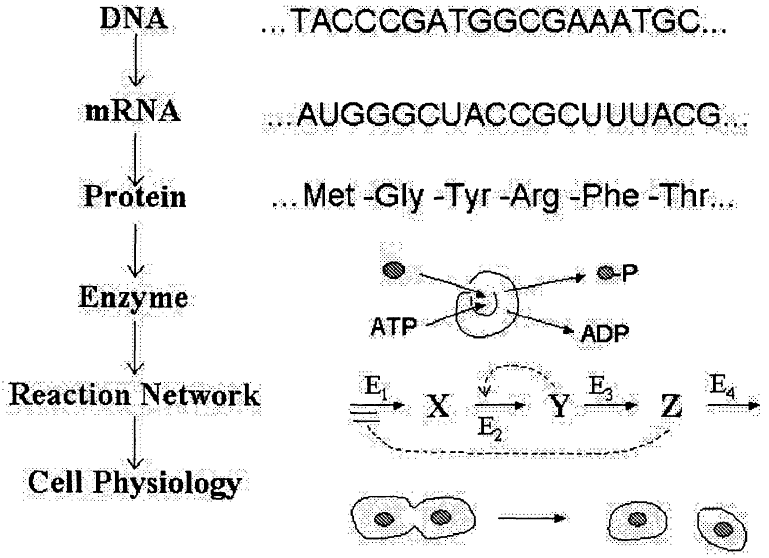


Figure 1. Bioinformatics research involves multiple modes of investigation. Alongside each level of abstraction, examples of data and physical entities are shown. Figure courtesy J.J. Tyson (Virginia Tech).

ciation, classification, and clustering techniques, the complexity of such abstraction is tremendous. Furthermore, the performance of these analysis techniques in the context of specific data sources may be suboptimal. Where such abstraction is possible, development of computationally efficient analysis techniques pose further challenges.

Consequently, in this chapter, we discuss analysis tasks in the emerging area of bioinformatics applications. We outline the nature of analysis tasks in these domains, formalize the tasks, and describe the state-of-the-art in algorithms for accomplishing them. We conclude with a discussion of ongoing research and future directions, both in terms of the tasks and the domains motivating their application.

2. Bioinformatics Applications

Bioinformatics, fueled by the information-centric view of biology, has ushered in a veritable revolution in genomics involving technologies such as DNA sequencing and linkage analysis. The recent completion of a rough map of the entire human genome (by Celera Genomics Inc., MD, USA and the federally-funded multi-nation Human Genome Project (HGP)) and the complete sequencing of the *Arabidopsis Thaliana* plant signify major milestones in data acquisition. A compelling frontier is the design of software tools that can harness and mine the continuing amount of data generated by such research efforts. Determining the structure of complex organic molecules, correlating the structure and function of the molecule, and engineering a molecule with desired function present key challenges in bioinformatics. The size and complexity of the molecules renders these tasks extremely difficult and computation-intensive. Fig 1 describes the multiple granularities at which bioinformatics research is typically conducted.

Problem Formulation

One of the widely studied problems in bioinformatics is the determination of primary structure involving DNA sequences. The rapid industrialization of this activity has almost exclusively focused on sequencing DNA fragments that correspond to the expressed sequence tags (ESTs) only (these might represent gene products that are involved in a specific activity, for instance). The exact location of the ESTs on the complete genome sequence is determined by the 'binding sites' and several parameters that capture the goodness-of-fit [SK99]. Various web databases now provide searching capabilities for genomes of already sequenced organisms. Many of them support BLAST searches that attempt a primitive level of information integration, in addition.

This current emphasis on elucidating the structure of the genome is referred to as *structural genomics*. Extensive research has also been done on computational techniques for correlating structure and function of the gene (*functional genomics*). These techniques span the areas of machine learning (clustering, classification), algorithmics (sequence alignments, pattern matching), and statistics (Bayesian learning, model fitting). An example of structure-function correlation is illustrated in the case of tumor protein $p53$ (TP_{53}). The human gene TP_{53} belongs to the family $p53$ of proteins. This family is responsible for suppressing the growth of tumors ('preventing cancerous mutiny' [Rid00]). Through extensive data analysis, it has been found that $p53$ is frequently mutated or inactivated in about 60% of cancers. Also, it has been found that $p53$ interacts with cancer-associated HPV ($e6$) viral proteins causing the degradation of $p53$. This requires an additional factor $e6 - ap$, which stably associates with $p53$ in the presence of HPV viral proteins.

In contrast to TP_{53} about which a fair deal is known, there are other proteins about which very little is known. For example, it is known that *BRC1_HUMAN* (Breast Cancer Type 1 Susceptibility Protein) is known to have high incidences of mutations in breast and ovarian

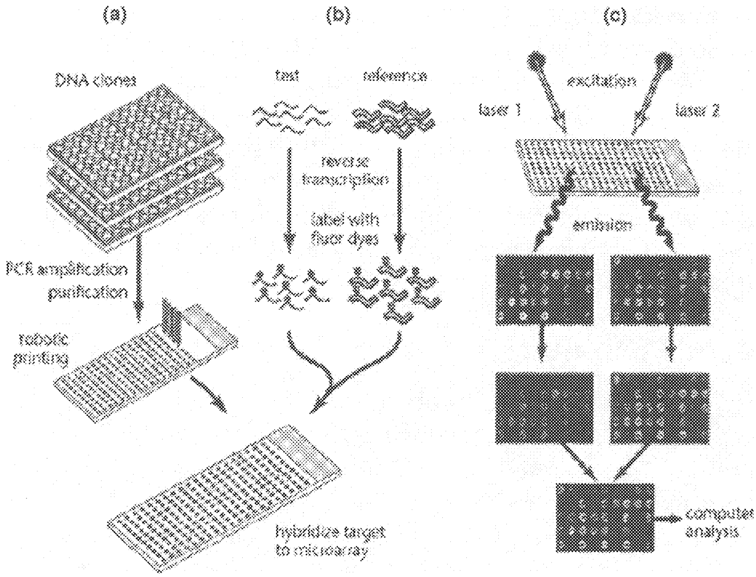

(a)
(b)
(c)
DNA clones
PCR amplification
purification
robotic printing
test
reference
reverse transcription
label with fluor dyes
hybridize target to microarray
excitation
laser 1
laser 2
emission
computer analysis

Figure 2. Schematic outline of a microarray design and analysis experiment [DBC$^+$99]. (a) Genes of interest are made available as DNA clones. A polymerase chain reaction (PCR) is used to amplify each gene to a sufficient amount to allow 'printing' onto the array. (b) RNA is isolated from reference (*control*) cells and test (*experimental*) cells. cDNA is synthesized from each RNA population by a reverse transcription process. The resulting two cDNA samples (control and experimental) are labeled with two different fluorescent dyes, mixed, and hybridized to the targets on the microarray. (c) Hybridization is monitored through measurement of the amount of fluorescence associated with individual target molecules on the array. Since the two dyes fluoresce at different wavelengths, it is possible to distinguish hybridization due to the control cDNA population from that due to the experimental cDNA population. Figure courtesy J.M. Trent (National Institutes of Health). Reprinted with permission of Nature Genetics.

cancer families (45% of inherited breast cancers). However, the precise function of this protein is not known, and is suspected to be responsible for regulating gene expression. Relatively little is known about the other proteins in this family. Clearly, the usefulness of this information is tremendous.

The ultimate goal of these endeavors is to gain an understanding of the underlying biological, molecular, and biochemical processes. From performing functional studies on genes or gene families in isolation, research has progressed to studying all the genes in a given organism simultaneously. A technique that has aided in this 'massive parallelization' of experimental biology is microarray bioinformatics.

Microarrays (sometimes referred to as *DNA chips*) form a relatively new technique in bioinformatics, inspired by miniaturization trends in micro-electronics. The basic idea is to 'print' DNA templates (*targets*), for all available genes that can be expressed in a given organism, onto a high-density 2D array in a very small area on a solid surface[1]. The goal then is to determine the genes that are expressed when cells are exposed to experimental conditions, such as drought, stress, or toxic chemicals. To accomplish this, RNA molecules (*probes*) are extracted from the exposed cells and 'transcribed' to form complementary DNA (cDNA) molecules. These molecules are then allowed to bind (*hybridize*) with the targets on the microarray and will adhere only with the locations on the array corresponding to their DNA templates. Typically such cDNA molecules are tagged with fluorescent dyes, so the expression pattern can be readily visualized as an image. Intensity differences in spots will then correspond to differences in expression levels for particular genes. Using this approach, one can 'measure transcripts from thousands of genes in a single afternoon' [HA00]. Microarrays thus constitute an approach of great economic and scientific importance, one whose methodologies are continually evolving to achieve higher value and to fit new uses. Applications utilizing microarrays are characterized by the need to couple experimental design with data-driven analyses, to integrate wet-lab pipetting, robotic manipulation, and image readers at the software level, and to track and identify data and physical entities as they proceed through the laboratory and computational pipeline. Fig. 2 depicts the processes and management of physical entities in a typical microarray experiment. An example image file from microarray analysis is shown in Fig. 3.

While the study of genes, their sequences, and expression has seen significant experimental and computational advances, the closely related area of proteomics offers tremendous potential for research. The term proteome [WWAHE97] refers to the study of proteins expressed by a genome. The basic questions in proteomics examine the structure of proteins and its correlation with function. This task bears tremendous complexity because of the combinatorics involved - aminoacids (roughly 100 of them, with 20 natural aminoacids and the rest, modified aminoacids) provide the building blocks for proteins. Proteins contain upwards of several thousand amino-acids in polypeptide bonds. There are many more proteins in a proteome than genes in a genome since there are various ways in which a gene can be spliced in constructing mRNA.

In medical conditions such as cancer, a combination of several genetic alterations are necessary. Detecting such networks of gene expressions (epigenetics) is an extremely expensive task. In theory, it is possible to

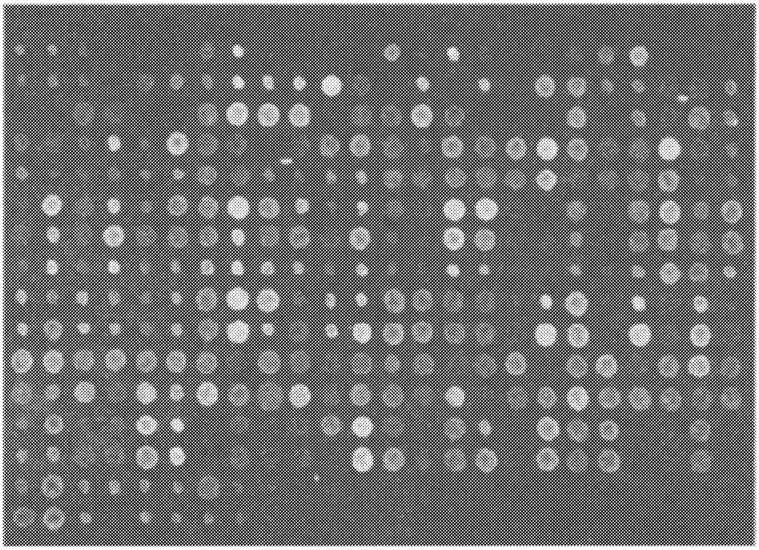


Figure 3. Biological variation of gene expression in Loblolly pine clones by microarray analysis. Data mining is used to identify clusters of genes that are co-expressed under similar stress/drought conditions, helping to hypothesize and/or update gene regulatory networks. The microarray is printed in four 24 × 16 sub-quadrants, one of which is shown here. Figure courtesy Y.-H. Sun (NCSU).

directly correlate genes to associated functionality (as opposed to examining the relationship between genes and proteins and proteins and function). A logical avenue for such investigation is to disrupt selected genes and study its effect. However, disrupting a single gene does not have simple consequences. Invariably, a single disruption will impact the expression of many genes, thus making a direct causality difficult to establish. Furthermore, recent studies have demonstrated a lack of good correlation between the presence of mRNA and associated protein amount in a given cell at a given time. All of these factors motivate the need for a concerted effort at the structural and functional characterization of proteins.

A final frontier in bioinformatics is the modeling of entire regulatory networks that are involved in processes such as cell division in mammalian systems, signal transduction, and drought resistance in plant systems. Modeling such phenomena involves capturing the complexity of gene interaction as well as levels of protein abundance (which correspond to the 'transcription' of genes). Protein abundance is typically measured using 2D Gel Electrophoresis that separates and identifies protein samples in an XY plane using displacements at right angles to determine gradients (see Fig. 4). The integration of this information in a complete regulatory pathway is currently a cumbersome manual process whereby molecular biologists, biochemists, and domain specialists painstakingly piece together known components to create a representation of the system as a whole. For instance, the online resource `www.biocarta.com` provides interactive clickable graphical models for various signaling and receptor pathways. Links to existing databases and online literature help associate references and annotations with pathways.

Steps Involved in Data Analysis

The primary tasks in data mining fall into three categories — design of metrics for effective sequencing, clustering and incorporation of biological knowledge into analysis, and information integration. The design of new comparison metrics is currently a black art involving finding prospective binding sites and assembling them to form the final sequence. The accepted practice is to perform multiple sequence alignments that identify potential discrepancies in insertions and deletions, genotype differences, and 'splicing errors.' It also supports a consensus approach to locating genes in genomic sequences. The underlying algorithms include dynamic programming, machine learning techniques (such as decision trees), and statistical Markov models.

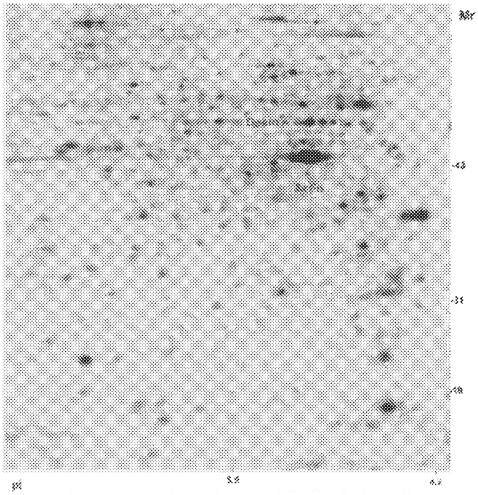


Figure 4. 2D Electrophoresis gel image of a human muscle. Figure courtesy Human 2D Page Database at `http://biobase.dk/cgi-bin/celis/`. Reproduced with permission.

Experimental procedures such as microarrays have firmly established the role of clustering as a key step in bioinformatics. The detection of gene expression changes (for varying levels of stress, say) coupled with modeling time and/or position effects is an area of active research in the bioinformatics community. Unfortunately, this problem domain is characterized by imperfections in the experimental process, lack of repeatability, and the high cost of obtaining experimental data. In addition, the sources and causes of experimental variability are not well-understood. As a result, a variety of preprocessing techniques including principal component analysis, feature handcrafting, data filtering and class-preserving projections are applied before clustering. In addition, clusters can occur in widely varying sizes, shapes, densities, and orientations. Purely similarity-based numeric clustering is often ineffective for categorical data and can cause information loss due to discretization, if applied without careful study.

With the rapid emergence of data formats and bioinformatics applications supporting a veritable cottage industry of databases, information integration becomes paramount. The diversity of originating sources such as protein banks, sequence databases and encyclopedias of pathways, make information integration an important and challenging problem. Information integration is typically addressed by remapping queries to originating sources, introducing a transparency layer of middleware between data sources, employing federated database schemas, or using

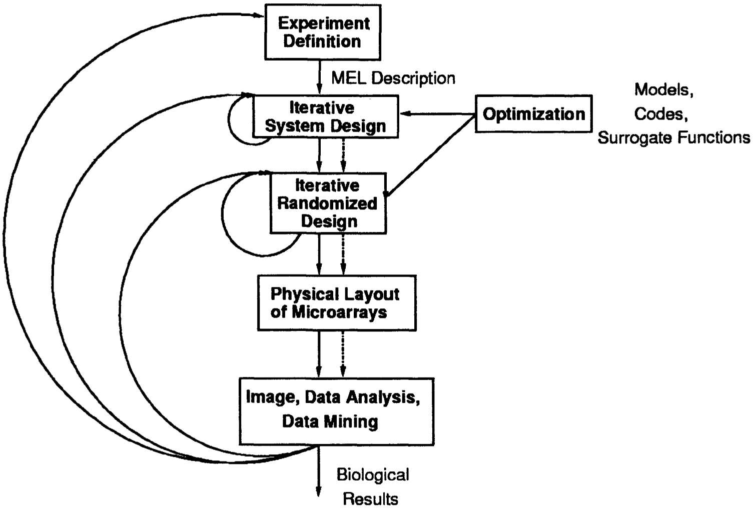

Experiment Definition
MEL Description
Iterative System Design
Optimization
Models,
Codes,
Surrogate Functions
Iterative Randomized Design
Physical Layout of Microarrays
Image, Data Analysis, Data Mining
Biological Results

Figure 5. Execution and flows in the *Expresso* microarray experiment management system [AHCR01]. The solid lines indicate computational flows; the dashed lines indicate realizations of the pipeline in devices such as robots and image readers.

other mediator-based schemes. Such simple approaches will be infeasible in the foreseeable future, due to the emergence of non-traditional forms of information content and rapidly evolving standards, characterizing semi-structured data. For example, the query 'Find all information pertaining to human chromosome 22' was considered an impossible query under the Human Genome Project (HGP) until just a few years back [BDH$^+$95]. While traditional information integration literature addresses information fusion from data types such as records, database tables [GMUW00], images, and hierarchies of information elements, large scale bioinformatics research calls for a more holistic approach to information integration [FKKR99].

Analysis steps such as clustering can also take into account apriori background knowledge, wherever available, for hypothesizing gene regulatory networks. Techniques such as inductive logic programming [Mug99] are especially appropriate here. Standardization of data formats, and information integration support are thus important for effective and seamless application of data mining at all stages in bioinformatics.

Successful Applications

A variety of integrated systems have been developed that support focused tasks in bioinformatics. For example, the REX program [YC98] performs integrated sequencing and clustering by extending rough assembled drafts. The Morgan system [Sal98] provides integrated support for analyzing eukaryotic and prokaryotic gene sequences. Software tools such as *Scanalyze* and *MicroArray Suite* [Inc00] provide facilities for image processing and basic statistical analysis of microarray experiments. Many of these tools also incorporate analysis techniques such as support-vector machines [BGL+00; FCD+00; Vap95], PCA [BDF+97], and bayesian analysis [Hec96]. The necessity of replication to obtain statistically significant results is also reflected in the software designs [LKWS00]. Even a cursory glance through literature in journals such as *Bioinformatics* reveals the role of data mining in experimental biology. Dedicated web resources have been developed for domains such as cyanobacteria [Ins96], eukaryotic cell cycle modeling, and plant genomics. Some of these sites even provide integrated sequencing, analysis, and visualization options. Some integrated systems provide facilities to intelligently close-the-loop in data analysis and acquisition, thus providing sophisticated capabilities for problem solving. For example, the *Expresso* microarray problem solving environment (PSE) (Fig. 5) utilizes computational models of various physical stages of microarray experiments to determine new parameters for experimental runs, and perform automated randomized placement on arrays, and planning of replication [AHCR01].

3. Future Issues

A close inspection of the above applications reveals various similarities and also an interesting dichotomy. The recurring theme is, of course, the emphasis on data and experiment management for supporting the lifecycle of computational simulation and modeling. The contrasts arise in the methodologies followed in various projects. For sequence analysis and structural genomics, standalone database systems now provide native support for domain-specific analyses The end goal is to tap an extensive commercial infrastructure to harness scientific data. To this end, various federal agencies now provide universal web support for scientists to explore high level queries such as "provide a list of Arabidopsis ESTs consistent with my sequences." Powerful query optimization algorithms have been developed [CS99; Hel98] that selectively 'push' costly imaging operations deeper (earlier) into the computational pipeline. A comprehensive effort of this nature that extends to simulation data (defined

```
EXPERIMENT PINE_DROUGHT_GROWTH May-August,2000 "384 clones"
...
DYE CY3 "Genisphere Kit"
DYE CY5 "Genisphere Kit"
...
PRINTING_ROBOT NCSU_FBC "Brown-type robot at NCSU"
...
PRINTING_CONFIGURATION Stanford4x16x24  4 16 24 QUADRANTS
PRINTING_CONFIGURATION Stanford4x22x24  4 22 24 QUADRANTS
...
TISSUE D4M        D4       Needles Unstressed (Control)
TISSUE D4I        D4       Needles Intermediate Stressed
```

Figure 6. Example description of a microarray experiment in the *Expresso* system [AHCR01] which includes the name of the experiment, the dyes used, description of the printing robot, printing formats, and the tissue configurations. The lines are in a self-describing format where the start field identifies the nature of information modeled.

over discretizations involving continuous data) and associated queries would be useful.

The other extreme, witnessed in large scale projects, is to provide experiment management systems that model the entire problem solving process and provide conceptual and visual abstractions for reasoning about the various stages in a typical experiment/simulation. Many bioinformatics tools are oriented toward cataloging descriptions of experiments, scenarios of information integration, and using these to manage the computational environment. Subsequently, a variety of formats (e.g., XML for microarray data) have been recently proposed. While many are primarily focused on the problems of syntactic data interchange, such markup languages are more important in associating metadata with various stages of bioinformatics experiments (see Fig. 6 for an example description from the *Expresso* system).

Nevertheless, large-scale simulations and experiments from both perspectives are increasingly generating massive datasets. One approach that might benefit future applications is that of compression of data to preserve interesting artifacts while reducing data volume. The problem in these cases becomes one of identifying appropriate distortion measures and approximation heuristics capable of preserving artifacts in data. In many cases, data reduction is also important for algorithms that have high computational costs.

New mining techniques also have to be developed, particularly algorithms that perform constructive induction along with generalization to arrive at succinct combinations of feature vectors. Craven and Shavlik

[CS93] showed that the problem of finding the right codon representation (the mapping from DNA sequences to amino acids) is notoriously difficult with traditional machine learning techniques. This feature will be especially useful in gene expression analysis where ESTs are assigned to putative categories based on their function. The need to combine such representational and mining issues will be critical in semi-automated sequencing and analysis activities.

Finally, a number of related efforts on distributed environments, novel computing paradigms (such as SETI used in astrophysics), advanced networking and visualization infrastructure indicate that data mining must become seamlessly integrated to solve grand challenge problems, such as multi-genome analysis and comparison. Incremental and anytime analysis techniques will contribute to the long-term success of integrating observations, simulations, and mining in bioinformatics.

Acknowledgments

The authors wish to thank L.S. Heath, R.G. Alscher, L.T. Watson, J.J. Tyson (Virginia Tech), Y.-H. Sun, R.R. Sederoff (NCSU) for helpful discussion and providing permission to use some of the figures described in this article.

Notes

1. 'Array' here refers merely to a rectangular arrangement; the 'axes' do not have any biological interpretation.

References

[AHCR01] R.G. Alscher, L.S. Heath, B.I. Chevone, and N. Ramakrishnan. Expresso - A PSE for Bioinformatics: Finding Answers with Microarray Technology. In A. Tentner, editor, *Proc. of the High Performance Computing Symposium, Advanced Simulation Technologies Conference*, pages 64–69, Seattle, WA, April 2001.

[BDF+97] D. Barbara, W. DuMouchel, C. Faloutsos, P. Haas, J. Hellerstein, Y. Ioannidis, H. Jagadish, T. Johnson, R. Ng, V. Poosala, K. Ross, and K. Sevcik. The New Jersey Data Reduction Report. *Bulletin of the IEEE Technical Committee on Data Engineering*, Vol. 20(4):pp. 3–45, December 1997.

[BDH+95] P. Buneman, S. Davidson, K. Hart, C. Overton, and L. Wong. A Data Transformation System for Biological

Data Sources. *Proc. of the VLDB Conference*, 1995.

[BGL$^+$00] M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares Jr., and D. Haussler. Knowledge-Based Analysis of Microarray Gene Expression Data by Using Support Vector Machines. *Proceedings of the National Academy of Science*, Vol. 97(1):pp. 262–267, January 2000.

[CBCR98] K.M. Chandy, R. Bramley, B.W. Char, and J.V.W. Reynders. Report of the NSF Workshop on Problem Solving Environments and Scientific IDEs for Knowledge, Information and Computing (SIDEKIC'98). Technical report, Los Alamos National Laboratory, 1998.

[CS93] M. Craven and J. Shavlik. Learning to Represent Codons: A Challenge Problem for Constructive Induction. *Proc. of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1319–1324, 1993. Chambery, France.

[CS99] S. Chaudhuri and K. Shim. Optimization of Queries with User-Defined Predicates. *ACM Transactions on Database Systems*, Vol. 24(2):pp. 177–228, June 1999.

[DBC$^+$99] D.J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J.M. Trent. Expression Profiling Using cDNA Microarrays. *Nature Genetics*, Vol. 21:pp. 10–14, 1999.

[FCD$^+$00] T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data. *Bioinformatics*, Vol. 16(10):pp. 906–914, 2000.

[FKKR99] D. Fensel, N. Kushmerick, C. Knoblock, and M.-C. Rousset. *Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration.* International Joint Conference on Artificial Intelligence, Stockholm, 1999.

[GMUW00] H. Garcia-Molina, J.D. Ullman, and J. Widom. *Database System Implementation.* Prentice Hall, 2000.

[HA00] H. Hamadeh and C.A. Afshari. Gene Chips and Functional Genomics. *American Scientist*, Vol. 88:pp. 508–515, November-December 2000.

[Hec96] D. Heckerman. Bayesian Networks for Knowledge Discovery. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and

R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 273–306. AAAI/MIT Press, 1996.

[Hel98] J.M. Hellerstein. Optimization Techniques for Queries with Expensive Methods. *ACM Transactions on Database Systems*, Vol. 23(2):pp. 113–157, September 1998.

[Inc00] Scanalytics Inc. Scanalytics Microarray Suite. `http://www.scanalytics.com`, 2000.

[Ins96] Kazusa DNA Research Institute. CyanoBase. URL: `http://www.kazusa.or.jp/cyanobase/`, 1996.

[LKWS00] M.-L. T. Lee, F.C. Kuo, G.A. Whitmore, and J. Sklar. Importance of Replication in Microarray Gene Expression Studies: Statistical Methods and Evidence from Repetitive cDNA Hybridizations. *Proceedings of the National Academy of Science*, Vol. 97(18):pp. 9834–9839, August 2000.

[MBM$^+$98] R.W. Moore, C. Baru, R. Marciano, A. Rajasekar, and M. Wan. Data-Intensive Computing. In C. Kesselman and I. Foster, editors, *The Grid: Blueprint for a New Computing Infrastructure*, chapter 5, pages 107–129. Morgan Kaufmann, 1998.

[MPE98] R.W. Moore, T.A. Prince, and M. Ellisman. Data-Intensive Computing and Digital Libraries. *Communications of the ACM*, Vol. 41(11):pp. 56–62, November 1998.

[Mug99] S. Muggleton. Scientific Knowledge Discovery Using Inductive Logic Programming. *Communications of the ACM*, Vol. 42(11):pp. 42–46, 1999.

[RB96] J.R. Rice and R.F. Boisvert. From Scientific Software Libraries to Problem-Solving Environments. *IEEE Computational Science & Engineering*, Vol. 3(3):pp. 44–53, Fall 1996.

[Rid00] M. Ridley. *Genome: The Autobiography of a Species in 23 Chapters*. Harpercollins, 2000.

[Sal98] S.L. Salzberg. Decision Trees and Markov Chains for Gene Finding. In S. L. Salzberg, D. B. Searls, and S. Kasif, editors, *Computational Methods in Molecular Biology*, pages 187–203. Elsevier, 1998.

[SK99] S. Schulze-Kremer. Disccovery in the Human Genome Project. *Communications of the ACM*, Vol. 42(11):pp. 62–64, November 1999.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[WWAHE97] M.R. Wilkins, K.L. Williams, R.D. Appel, and D.F. Hochstrasser (Eds.). *Proteome Research: New Frontiers in Functional Genomics (Principles and Practice).* Springer Verlag, 1997.

[YC98] D.P. Yee and D. Conklin. Automated Clustering and Assembly of Large EST Collections. *Proc. of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 1998.

Chapter 9

MINING RESIDUE CONTACTS IN PROTEINS

Mohammed J. Zaki and Chris Bystroff

Abstract In this paper we develop data mining techniques to predict 3D contact potentials among protein residues (or amino acids) based on the hierarchical nucleation-propagation model of protein folding. We apply a hybrid approach, using a Hidden Markov Model to extract folding initiation sites, and then apply association mining to discover contact potentials. The new hybrid approach achieves accuracy results better than those reported previously.

Keywords: Protein Structure Prediction, Contact Maps, Data Mining, Hidden Markov Models, Association Rules, Protein Folding

1. Introduction

Today we are witnessing a paradigm shift in predicting protein structure from its known amino acid sequence $(a_1, a_2, \cdots, a_n)$. The traditional or *ab initio* folding method employed first principles to derive the 3D structure of proteins. However, even though considerable progress has been made in understanding the chemistry and biology of folding, the success of *ab initio* folding has been quite limited.

Instead of simulation studies, an alternative approach (used in this paper) is to employ learning from examples using a database of known protein structures. For example, the Brookhaven Protein Database (PDB) records the 3D coordinates of the atoms of thousands of protein structures. Most of these proteins cluster into around 700 fold-families based on their similarity. It is conjectured that there will be on the order of 1000 fold-families for the natural proteins [WGK00]. The PDB thus offers a new paradigm to protein structure prediction by employing data mining methods like clustering, classification, association rules, hidden Markov models, etc.

A fascinating property of protein chains is that they spontaneously and reproducibly fold themselves into complex three-dimensional globules when

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 141–164.
© 2001 *Kluwer Academic Publishers.*

placed in an aqueous solution. The sequence of amino acids making up the polypeptide chain contains, encoded within it, the complete building instructions. This self-organization cannot occur by a random conformational search for the lowest energy state [Hon99], since such a search would take millions of years, while proteins fold in milliseconds. In recent years, a combination of molecular biological and biophysical techniques have dissected the folding process into fast and slow components which localize to certain parts of the amino acid sequence [SMF92].

Some small, fast-folding regions of the molecule may be identified by their sequence alone. A library of short sequence patterns that fold fast has been compiled by cluster analysis of the database of known protein structures (the I-sites Library, [BB98]). In this work, similar short sequences that mapped to the same local structure in different proteins were deemed to be autonomous folding units, and the short sequences were compiled into patterns or "profiles" which could then be used to predict whether or not a segment of the protein would tend to fold independently of the rest of the molecule. Cross-validation showed a strong statistical significance to the predictions made by the profiles, and later NMR studies showed that some peptides predicted to fold in isolation actually did so [YBR$^+$98]. Peptides with a strong tendency to fold independently constitute about 30% of the amino acid residues in protein sequences. The formation of independent folding units (I-sites motifs) is the first level of self-organization in the folding process: the "initiation."

These short motifs occur in proteins of widely differing topology, and so cannot contain sufficient information to define the overall, global fold of the protein molecule. Moreover, they are too short to be the fast-folding regions found by experimental dissection. There must be a higher level of self-organization which dictates how the short pieces come together to form larger, longer globular domains. The rules defining the propagation of structure along the chain, starting from the sites of initiation, have been extracted from the database of known protein structures and formalized as a hidden Markov model (HMM), called HMMSTR [BTB00] (or "hamster"), discussed further below. HMMSTR models the interactions between adjacent short regions of the sequence, and so attempts to model the second level of self-organization: "propagation" of structure along the sequence.

The I-sites Library models the initiation sites of folding, and the new HMM models interactions between those sites. But HMMSTR [BTB00] is a network of connections between I-sites motifs, and thus simultaneously models both folding initiation and propagation. The two levels of complexity, not discretely defined but smoothly intermingled, are represented in the HMM as variable degrees of branching. Unbranched segments are initiations sites, whose probabilities depend simultaneously on short contiguous segments of the sequence, while branching and cycles represent multiple sequence-dependent ways of extending and linking the initiation sites. Arbitrary levels of complexity may be modeled by including HMMs recursively within overarching HMMs, the latter representing the ways of connecting the output of the HMMs it contains. Hidden Markov models are limited to data that can be expressed as one-dimensional sequences of discrete symbols, but there are techniques for overcoming both the discreteness and the one-dimensionality [Rab89].

The next level of complexity in protein folding is called "condensation". In the first few microseconds after introducing the polypeptide chain into an aqueous solution, initiation sites form transient, rapidly-interchanging structures, favoring one or more 3D conformations to varying degrees. These structures propagate along the chain by promoting compatible upstream and downstream conformations, and the resulting transiently-formed substructures encounter each other by through-space diffusion, condensing into larger, ordered globules, as energy dictates. The ordering of these three processes is not discrete but overlapping, and they should therefore be integrated into a single computational model. Modeling of the condensation step given predictions based on the modeling of initiation/propagation is the subject of the present work. A single Markov state prediction implies a local substructure and a single amino acid position within it. Thus, a contact between two Markov states implies a specific mode of condensation between two local substructures to form tertiary structure.

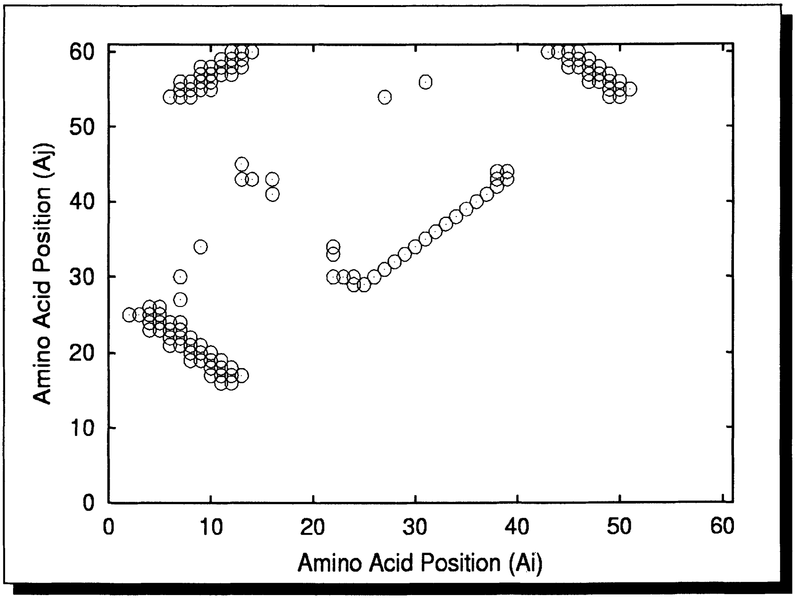


Figure 1. Contact Map (PDB protein 2igd, length 61). Circles indicate residue contacts, and clusters of circles indicate certain secondary structures.

The contact map of a protein (see Figure 1) is a particularly useful representation of protein tertiary structure. Two amino acids in a protein that come into contact with each other form a non-covalent interaction (hydrogen-bonds, hydrophobic effect, etc.). More formally, we say that two residues (or amino acids) a_i and a_j in a protein are in *contact* if the 3D distance $\delta(a_i, a_j)$ is less than some threshold value t (in this paper we use $t = 7\mathring{A}$ as the threshold distance), where $\delta(a_i, a_j) = |\mathbf{r_i} - \mathbf{r_j}|$, and $\mathbf{r_i}$ and $\mathbf{r_j}$ are the coordinates of the α-Carbon atoms of amino acids a_i and a_j. We define *sequence separation* as the distance between two amino acids a_i and a_j in the amino acid sequence, given as $|i - j|$. A contact map for a protein with N residues is an $N \times N$ binary matrix C whose element $C(i, j) = 1$ if residues i and j are in contact, and $C(i, j) = 0$ otherwise. The contact map provides a host of useful information. For example, secondary structure can easily be discerned from it. α-Helices appear as thick bands along the main diagonal since they involve contacts between one amino acid and its four successors, while β-Sheets are thin bands parallel or anti-parallel to the main diagonal, etc. However, tertiary structure is not easily found from the contact map. For predicting the elusive global fold of a protein we are usually interested in only those contacts that are far from the main diagonal. In this paper we thus ignore any pair of residues whose sequence separation $|i - j| < 4$.

Previous work on contact prediction has employed neural networks [FC99], and statistical techniques based on correlated mutations [OV97, TCS96]. Recent work by Vendruscolo et al [VKD97] has also shown that it is possible to recover the 3D structure from even corrupted contact maps. In this paper we present a new hybrid technique for contact map prediction. We first predict local structural elements using an HMM. The HMM simultaneously represents the initiation and propagation steps of protein folding. We then apply an association mining technique on top of the HMM states to predict the set of states that frequently co-occur with contacts. These sets are then used for predicting contacts in unseen proteins. Our model obtains 19% accuracy and coverage over the set of all proteins; the model is also 5.2 times better than a random predictor. We can significantly enhance coverage to over 40% if we sacrifice accuracy (13%). For short proteins (length< 100) we get 30% accuracy and coverage (4.5 times better than random); if we lower accuracy to 26% we can get coverage upto 63%. We believe that these results are better than (or equal to) those reported previously.

2. Hybrid Mining Approach

Here we describe the hybrid technique used to predict residue contacts. We first use an HMM to predict local substructures within the protein. We then use meta-level mining on the output of the HMM using association rule mining. The following sections provide a brief introduction to these two methods (readers familiar with them can safely skip ahead to Section 3).

2.1. Hidden Markov Models

The description of HMM below is based on the excellent tutorial by Rabiner [Rab89]. An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of observed symbols.

An HMM is made up of a finite number N of states. At each time step t a new state is entered based on a transition probability distribution which depends on the previous state (the Markovian property). After each transition is made, an observation output is produced according to a fixed probability distribution which depends on the the current state. Thus there a N such observation probability distributions.

As an example of modeling proteins via HMMs, let consider an "urn and residue" model. There are N urns (or states) each filled with a large number of the 20 possible amino acids. The observation sequence (a protein?) is generated by initially choosing one of the N urns (according to an initial probability distribution), selecting a residue from the initial urn, recording which amino acid it is, replacing it, and then choosing a new urn (state) according to a transition probability distribution associated with the current urn. A step corresponds to a residue position. Thus a typical observation sequence might be:

step or position	1 2 3 4 $\cdots$ T
urn (hidden) state	$q_3 q_1 q_1 q_2 \cdots q_{N-2}$
amino acid (observation)	G L A K $\cdots$ S

An HMM is made up of the following components: T is the length of the observation sequence; N the number of states in the model; M the number of observation symbols (for simplicity we assume here that the output is a discrete symbol, e.g. an amino acid. However we actually use a continuous vector output as we shall see later); $Q = \{q_1, q_2, \cdots q_N\}$ is set of HMM states; $V = \{v_1, v_2, \cdots, v_M\}$ is the set of output symbols; $A = \{a_{ij}\}$ gives the set of state transition probabilities, i.e., $a_{ij} = P(q_j \ at \ (t+1)|q_i \ at \ t)$; $B = \{b_j(k)\}$ is the output symbol probability distribution in state q_j, i.e., $b_j(k) = P(v_k \ at \ t|q_j \ at \ t)$; and finally $\pi = \{\pi_j\}$ gives the initial state distribution, i.e., $\pi_j = P(q_j \ at \ t = 1)$.

Using the model, an observation sequence $O = O_1 O_2 \cdots O_T$ is generated as follows: 1) choose an initial state i_1 based on π, 2) set position $t = 1$, 3) choose O_t according to $b_{i_t}(k)$, 4) choose i_{t+1} according to $\{a_{i_t i_{t+1}}\}$, $i_{t+1} =$

$1, 2, \cdots N$, and 5) set $t = t + 1$; return to step 3 if $t < T$; otherwise terminate the procedure.

An HMM can be compactly represented using the notation $\lambda = (A, B, \pi)$. There are three key problems that have to be solved to build a useful model: 1) *Evaluation Problem:* Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model $\lambda = (A, B, \pi)$, how to compute the probability of the observed sequence $P(O|\lambda)$. This can be solved using the Forward-Backward algorithm [Rab89]. 2) *Estimation Problem:* Given the observation sequence $O = O_1 O_2 \cdots O_T$, how to choose a state sequence $I = i_1 i_2 \cdots i_T$, which is optimal in some meaningful sense. This can be solved using the Viterbi algorithm [Rab89]. 3) *Maximization Problem:* How to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$. This can be solved using the Baum-Welch reestimation method [Rab89]. We will discuss in the next section the exact details of how the HMM is built to model proteins.

2.2. Association Rules

Since its introduction, Association Rule Mining (ARM) [?] has become one of the core data mining tasks, and has attracted tremendous interest among data mining researchers and practitioners. ARM is an undirected or unsupervised data mining technique, which works on variable length data, and it produces clear and understandable results. It has an elegantly simple problem statement, that is, to find the set of all subsets of items or attributes that frequently occur in many database records or examples, and additionally, to extract the rules telling us how a subset of items influences the presence of another subset.

The association mining task can be stated as follows: Let $\mathcal{I}$ be a set of items, and $\mathcal{D}$ a database of examples, where each example has a unique identifier (*tid*) and contains a set of items. A set of items is also called an *itemset*. An itemset with k items is called a k-itemset. The *support* of an itemset X, denoted $\sigma(X, \mathcal{D})$, is the number of examples in $\mathcal{D}$ where it occurs as a subset. An itemset is *frequent* or *large* if its support is more than a user-specified *minimum support (min_sup)* value.

An *association rule* is an expression $A \Rightarrow B$, where A and B are itemsets. The support of the rule is the joint probability of a example containing both A and B, and is given as $\sigma(A \cup B)$. The *confidence* of the rule is the conditional probability that an example contains B, given that it contains A, and is given as $\sigma(A \cup B)/\sigma(A)$. A rule is *frequent* if its support is greater than *min_sup*, and it is *strong* if its confidence is more than a user-specified *minimum confidence (min_conf)*.

The data mining task is to generate all association rules in the database, which have a support greater than *min_sup*, i.e., the rules are frequent, and which also have confidence greater than *min_conf*, i.e., the rules are strong. In this paper we are interested in rules with a specific item, called the *class*, as a consequent, i.e., we mine rules of the form $A \Rightarrow \mathbf{c_i}$ where $\mathbf{c_i}$ is a class attribute $(1 \leq i \leq k)$.

This task can be broken into two steps:

1 Find all frequent itemsets having minimum support for at least one class $\mathbf{c_i}$. The search space for enumeration of all frequent itemsets is 2^m, which is exponential in m, the number of items. However, if we assume that there is a bound on the example length, we can show that ARM is essentially linear in the database size [?].

2 Generate strong rules having minimum confidence, from the frequent itemsets. We generate and test the confidence of all rules of the form $X \Rightarrow \mathbf{c_i}$, where X is frequent.

2.2.1 Mining Frequent Closed Itemsets.

We mine the frequent sets based on the Formal Concept Analysis approach [?], which is a very elegant mathematical framework for extracting "concepts" from databases.

Consider an itemset X. Let $Y = \{E \in \mathcal{D} | X \subseteq E\}$ be the set of all examples E in the database $\mathcal{D}$ where X occurs. Further let $X' = \{i \in \mathcal{I} | i \in \cap_{E \in Y} E\}$ be the set of all items that are common to all examples in the set Y. Then we say that X is *closed* if $X = X'$. In other words X is the maximal set of items that is common to all examples in Y. A closed itemset is also called a *concept*.

The set of all closed frequent itemsets can be orders of magnitude smaller than the set of all frequent itemsets, especially for real (dense) datasets [?]. At the same time, we don't lose any information; the closed itemsets uniquely determine the set of all frequent itemsets and their *exact* frequency. Thus instead of mining all the frequent itemsets we only mine the frequent closed itemsets using the CHARM algorithm [?] we recently developed. A detailed description of the algorithm is beyond the scope of this paper. Suffice it to say that CHARM can handle very large disk-resident or external memory databases; it has been tested on databases with millions of examples, and it scales linearly in the database size. We refer the reader to [?] for the algorithm description and its efficiency.

3. HMMSTR: An HMM for local structure in proteins

We describe here the hidden Markov model, HMMSTR [BTB00], for general protein sequences based on the I-sites library of sequence-structure motifs [BB98]. In the next section we will show how we apply association mining on the output of HMMSTR to predict residue contacts.

Unlike the linear HMMs used to model individual protein families [Edd98], HMMSTR has a highly branched topology and captures recurrent local features of protein sequences and structures that transcend protein family boundaries. The model extends the I-sites library by describing the adjacencies of different sequence-structure motifs as observed in the protein database, and achieves a great reduction in parameters by representing overlapping motifs in a much more compact form.

The I-sites (Invariant or Initiation sites) library consists of an extensive set of short sequence motifs, length 3 to 19, obtained by exhaustive clustering of sequence segments from a non-redundant database of known structures [HB96, BB98]. Each sequence pattern correlates strongly with a recurrent local structural motif in proteins. Approximately one third of all residues in the database are found in an I-sites motif that can be predicted with a high degree of confidence ($> 70\%$). The library is non-redundant in that no motif is completely contained within another, longer motif. However, many of the motifs overlap. Furthermore, the isolated motif model does not capture higher order relationships such as the distinctly non-random transition frequencies between the different motifs. The redundancy inherent in the I-sites model suggests a better representation that would model the diversity of the motifs and their higher order relationships while condensing features they have in common. A hidden Markov model is well suited to this purpose.

3.1. Description of HMMSTR

Each of the 262 I-sites motifs was represented as a chain of Markov states, each of which contains information about the sequence and structure attributes of a single position in the motif. Adjacent positions were represented by transitions from one state to the next. Hierarchical merging of these linear chains of states, based on sequence and structure similarity, resulted in a graph containing almost all the motifs. The merged graph of I-sites motifs comprises a network of states connected by probabilistic transitions, or more specifically, an HMM as shown in Figure 2.

Each state in HMMSTR can produce, or "emit", amino acids and structure symbols according to a probability distribution specific to that state. There are four probability distributions defined for the states in HMMSTR, b, d, r, and c, which describe the probability of observing a particular amino acid, secondary structure, backbone angle region, or structural context descriptor, respectively. A context descriptor represents the classification of a secondary structure type according to its context. For example, a hairpin turn is distinguished from a diverging turn, and a beta-strand in the middle of a sheet is distinguished from one at the end of a sheet. More formally, for a given state q_i, there are a set of emission probabilities, collectively called B_i. Here, we use four in this collec-

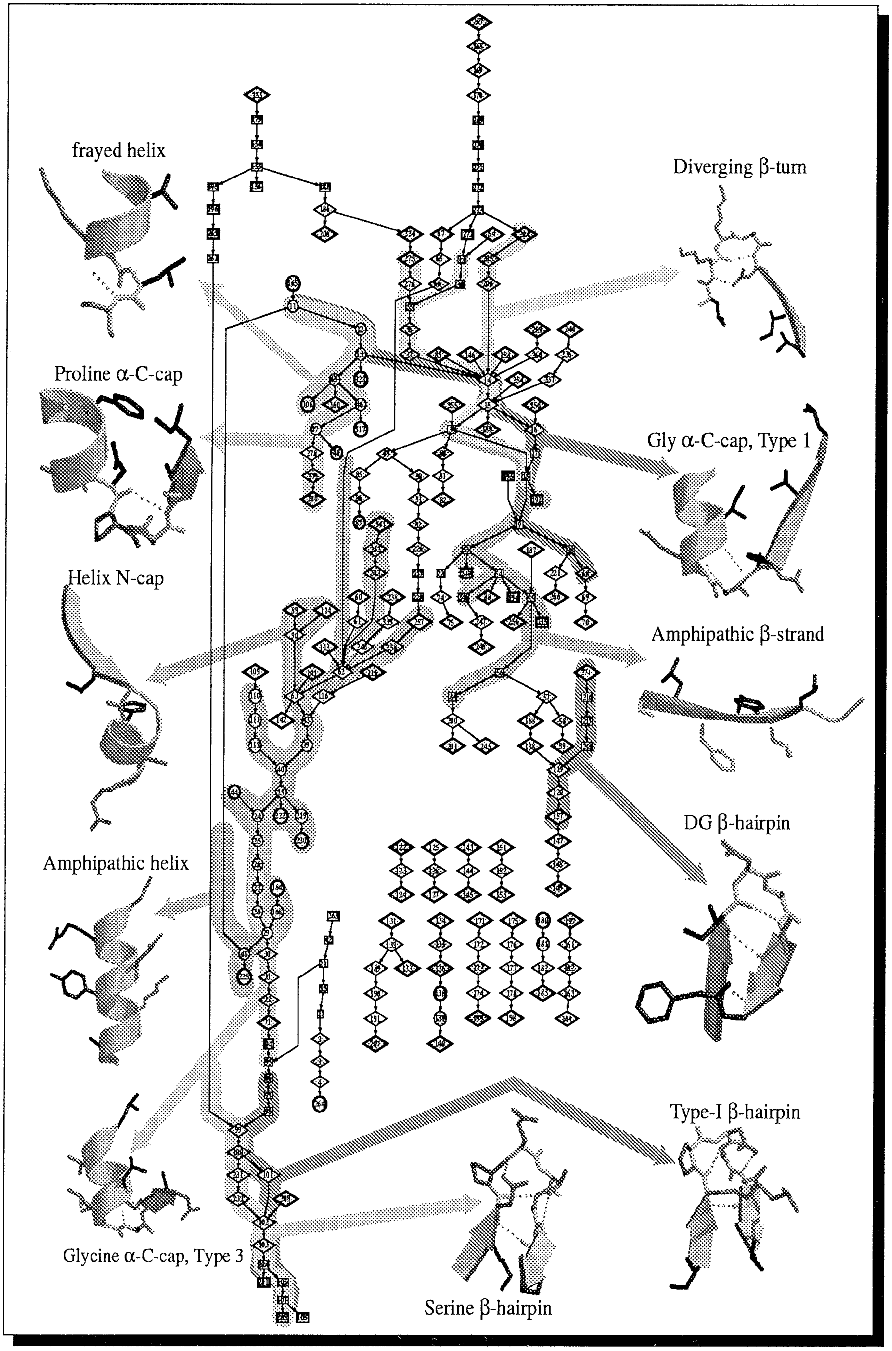

frayed helix
Diverging β-turn
Proline α-C-cap
Gly α-C-cap, Type 1
Helix N-cap
Amphipathic β-strand
DG β-hairpin
Amphipathic helix
Type-I β-hairpin
Glycine α-C-cap, Type 3
Serine β-hairpin

Figure 2. HMMSTR model built from I-sites Library. For a detailed description, see [BTB00]

tion, denoted *b*, *d*, *r*, and *c*. The values $b_i(m)$ $(1 \leq m \leq 20)$ are associated with probabilities for the emission of amino acids. The values $d_i(m)$ $(1 \leq m \leq 3)$, are the probabilities of emitting helix(H), strand(S) or loop/coil(T), respectively. The values $r_i(m)$ $(1 \leq m \leq 11)$ are the probabilities of emitting one of the 11 dihedral angle symbols. Finally, $c_i(m)$ $(1 \leq m \leq 10)$ are probabilities of emitting one of ten structural context symbols.

The database is encoded as a linear sequence of amino acids and structural observables. The amino acid sequence data consists of a "parent" amino acid sequence of known three-dimensional structure, and an amino acid profile obtained by alignments to the parent sequence [BB98]. The amino acid of the parent sequence is denoted by O_t, and the profile by $\{O_t^m\}(1 \leq m \leq 20)$. For the structural identifiers at each position t, the following nomenclature is used: 3-state secondary structure D_t, discrete backbone angle region R_t, and the context symbol C_t. A sequence s of length T is given by the values of the attributes at all positions $s_t = \{O_t, \{O_t^m\}, D_t, R_t, C_t\}$ $(1 \leq t \leq T)$. The utility of the HMM to model database sequences is based on the notion of a path. A path is a sequence of states through the HMM, denoted $Q = q_1 q_2 \cdots q_T$. Thus, the probability of a sequence s given the model λ, $P(s|\lambda)$, is obtained by summing the relevant contributions from all possible paths Q:

$$P(s|\lambda) = \sum_{all\ I} \pi_{i_1} B_{i_1}(s_1) a_{i_1 i_2} b_{i_2}(s_2) \cdots a_{i_{T-1} i_T} b_{i_T}(s_T)$$

where $I = i_1 i_2 \cdots i_T$ is a fixed sequence of states and $B_i(s_t)$ is the probability of observing s_t at state q_i, which for observation of a single sequence is given by

$$B_i(s_t) = \begin{pmatrix} d_i(D_t) \\ r_i(R_t) \\ c_i(C_t) \end{pmatrix} b_i(O_t)$$

Usually, only one of the structural emission symbols d, r, or c is included in B_i in any given training run. However, in principle, any combination could be used. Our HMMs showed significant improvements in performance when we used amino acid profiles instead of single amino acid sequences for training and for subsequent predictions. For the probability of observing a given profile $\{O_t^m\}$ position t in a sequence, we use the multinomial distribution, and the expression for B becomes

$$B_i(s_t) = \begin{pmatrix} d_i(D_t) \\ r_i(R_t) \\ c_i(C_t) \end{pmatrix} \sum_{m=1}^{20} b_i(m)^{Ncount \times O_t^m}$$

To give equal weight to the information in sequence families of different depths, $Ncount$ was taken to be a global parameter. $Ncount$ is the effective weight for the amino acid sequence profile b versus the other emission symbols (r, d, c). Statistically it should be the number of independent observations of the sequence in the database of proteins, which would be different for each protein, but we made it a global constant (i.e. the same for all proteins) so that no protein or set of proteins dominated the model. The value we ended up using (1.5) was arrived at by comparing parallel runs of expectation-maximization, each with a different $Ncount$ value.

3.2. Training HMMSTR

For training, evaluation and testing of the HMMSTR we used a non-redundant database of proteins of known structure, PDBselect:December 1998 [HS94], containing 691 proteins and their sequence families. The proteins in the set have $< 25\%$ sequence similarity. Entries in the database were selectively removed if the structure was solved by NMR, had a large number of disulfide bridges or cis-peptide bonds, or if it was a membrane-associated protein according to the header records. Disordered or missing coordinates in the middle of a sequence were addressed by dividing the sequence at that point. Contiguous segments of length less than 20 were ignored. Multiple sequence alignments were generated from each sequence using PSI-BLAST [AMS$^+$97] after filtering the query sequences for low-complexity regions [WF96]. Data for training the HMM included the sequence profile, computed from the multiple sequence alignment as described before [BB98], the DSSP secondary structure assignments [KS83], the backbone angles, and a structural "context" symbol.

Backbone angles were measured from the coordinates and assigned, using a Voronoi method, to 11 regions of phi/psi space. The centroids of 10 regions were chosen by K-means clustering of a large subset of trans phi/psi pairs from the database. The 11th region is all cis peptides.

A randomly selected set of 73 of the 691 proteins (19,000 positions) was then set aside and not used for training, but only for the final cross-validation. Before cross-validation, a test for true independence (compared to the training set) was applied to each member of the test set, and 12 members were removed. The final test set thus contained 61 proteins and 16,000 positions.

The remaining set of 618 parent sequences (145,000 positions) was used for training, and divided into a large set of 564 sequences (133,000 positions), used for optimization via the Expectation-Maximization algorithm, and a small set of 54 sequences (12,000 positions) used to evaluate the predictive ability of the model during training. Note that the small set of 54 sequences is used only for evaluation of the performance of a model and may thus appear to be a test set. However, decisions regarding the modification of the model are based on results of those evaluations. The set of 54 sequences is therefore not a test set, but a training set. For the final round of training we re-combined the large and small training sets, to a total of 618 sequence families. After the final round of training, the models were frozen.

4. Data Format and Preparation

After HMMSTR is built we again took the 691 proteins from PDBSelect and computed for each protein the optimal HMMSTR states that agree with the observed amino acids in the protein. In other words for each protein sequence-structure we solve the estimation problem, i.e., given the observation sequence $O = O_1O_2 \cdots O_T$, how to choose a state sequence $I = i_1i_2 \cdots i_T$, which is optimal. The output probability distributions of all the states thus chosen for a protein sequence is used as input for the association mining algorithm. In fact, rather than a single state associated with a given residue, we have available the probability that the residue at the given position is associated with all the states of HMMSTR, i.e., we have available $P(q_i|a_j)$ for all the 282 HMMSTR states $(1 \leq i \leq 282)$ for all the residues in a given protein $(1 \leq j \leq n$, where n is the length of the protein). For each residue we also know the amino acid at that position; the b, d, r, and c outputs, which describe the probability of observing a particular amino acid, secondary structure, backbone angle region, or structural context descriptor, respectively; the spatial coordinates of the α-Carbon atom $\langle x, y, z \rangle$; a distance vector of length n giving the distance of this residue from all other residues in the protein; and the 20 amino acid profiles for that position. A protein data file may look like this:

```
PDB Name: 1531_
Sequence Length: 185

Position: 1
Residue: R
Coordinates: 0.0  -73.2  177.6
Profile: 0.0 ... 1.0 ... 0.0 #20 Values
HMMSTR State Probabilities:
        0.0 ... 0.7 .... 0.3 ... 0.0 #282 Values
Distance Vector: 0 3 5 ... 18 15 13 #185 Values (Seq Length)

Position: 2
Residue: T
Coordinates: -124.4  0.2  -177.1
Profile: 0.0 ... 1.0 ... 0.0 #20 Values
HMMSTR State Probabilities:
        0.0 ... 0.9 ... 0.1 ... 0.0 #282 Values
Distance Vector: 3 0 3 ... 15 13 10 #185 Values

...
```

```
Position: 185
Residue: Y
Coordinates: -88.7  0.0  0.0
Profile: 0.0 ... 0.4 ... 0.6 ... 0.0 #20 Values
HMMSTR State Probabilities:
        0.0 ... 0.2 ... 0.5 ... 0.3 ... 0.0 #282 Values
Distance Vector: 15 13 10 ... 5 3 0 #185 Values
```

We have a file like the one shown above for all of the 691 non-redundant set of proteins from PDBSelect. Disordered or missing coordinates in the middle of a protein sequence were addressed by dividing the sequence at that point. This produces a set of 794 files, most of them containing an entire protein sequence, but some of these correspond to proteins that were split.

Given a protein file, we now have to transform the data into a format that can be easily mined for frequent closed itemsets, i.e., we need to transform the data into the relational or tabular format where we have multiple attributes (columns) for each example (rows) or record. Since we are interested in predicting the contact between a pair of amino acids, we use each pair as an example in the training set, associated with a special *class* attribute indicating whether it is a contact (C) or non-contact (NC); amino acids a_i and a_j are in contact if $\delta(a_i, a_j) < 7Å$, i.e., the distance between α-carbons of amino acids a_i and a_j is less then $7Å$. Our new database has an entry showing the two amino acids and their class for each pair of amino acids for each protein. In order to avoid predicting purely local contacts we ignore all pairs whose sequence separation $|i - j| < 4$. Note also that the number of contacts N_C is a lot smaller than the number of non-contacts N_{NC} for any protein.

We found that the percentage of contacts (or number of database entries with class 1) over all pairs is less than 1.7%. Across the 794 files, the longest sequence had length 907, while the smallest had length 35. There were 17,618,115 pairs over all proteins, while only 292,126 pairs were in contact. This database thus corresponds to a highly biased binary classification problem. That is, we have to build a mining model that can discriminate between contacts and non-contacts between amino acids pairs, where the examples are overwhelmingly biased towards the non-contacts.

Our database so far doesn't have enough information for good discrimination. All we have is the amino acids making up the pair and whether they are in contact or not. We need to add more "context" information to facilitate the classification. It is easy to incorporate, for each amino acid in the pair, the 3 secondary structure symbols (d_i, d_j), the 11 backbone angle regions (r_i, r_j), and the 10 structural context descriptors (c_i, c_j). For each pair we would also like to add the HMMSTR state probabilities. Since association rules only work for categorical attributes, we need to convert the continuous state probabilities into discrete values. To do this we take the ratio of each of the 282 HMMSTR state probabilities for a_i against the background or prior probability of an amino acid being in that state; if the ratio is more than some threshold we include the state in the context of a_i, else we ignore it. We repeat the same process for a_j. Using a similar thresholding method one can incorporate the amino acid profiles for positions i and j. With all this context information for both a_i and a_j we obtain a new database to be used to find the frequent itemsets characterizing the contacts and non-contacts. In summary the database might have the following columns for pairs of amino acids over all proteins:

```
Protein and Position Information:
                     ProteinID  PairID i j |i-j|
Amino Acids and Context: ai aj di dj ri rj ci cj
Profile: pi1 pi2 ... pj1 pj2 ...
HMMSTR: qi1 qi2 ... qj1 qj2 ...
Class: C or NC
```

Note that the number of columns can be variable for different pairs depending on the profile and HMMSTR state probabilities. p_{i_1}, p_{i_2}, etc. show the other amino acids that can appear in position i (provided the probability is more than some threshold), and finally q_{i_1}, q_{i_2}, etc. show HMMSTR states with probabilities more than some factor of the prior probability of those states.

5. Association Mining on the Pairs Database

We are now in a position to cast the above database in the association framework. Each attribute-value pair is an item, and is represented with a fixed, unique integer. For example $a_i = G$ is one item and $a_i = L$ is another item. By the same token each value of c_i, d_i, r_i, L_i, and R_i is a different item. Each of the HMMSTR states becomes a distinct item, as do the profile values. The items for the context attributes of a_i and a_j are also kept distinct. Finally we separate the examples that are contacts from those that are non-contacts to get two databases, denoted as $\mathcal{D}_C$ and $\mathcal{D}_{NC}$, respectively.

Given these databases our goal is to find high support and high confidence rules of the form $A \Rightarrow C$ and $A \Rightarrow NC$, that discriminate between the contact pairs and the non-contact pairs, respectively. Below we describe the mining/training and testing phases, where we learn from examples using the frequent closed itemsets, and then classify unseen examples as being contacts or non-contacts, respectively.

5.1. Mining on Known Examples

The goal of the mining phase is to learn from known contact and non-contact examples and build a model or rule set that discriminates between the two classes. We selected a random 90% of the files for training, out of a total of 794 files. The remaining 10% of the files were kept aside for testing the mined rule set.

Since we are primarily interested in predicting the contacts rather than the non-contacts, we mine only on the contacts database $\mathcal{D}_C$. However, we do use the non-contacts database $\mathcal{D}_{NC}$ to prune out those itemsets that are frequent in both sets. Building a discriminative rule set consists of the following steps, in order:

1. *Mining*: We use CHARM [?] to mine all the frequent closed itemsets in $\mathcal{D}_C$ based on a suitably chosen *min_sup* value (determined empirically). Let's denote the set of frequent closed itemsets as $\mathcal{F}$.

2. *Counting*: We compute the support of all itemsets in $\mathcal{F}$ in the non-contacts database $\mathcal{D}_{NC}$.

3. *Pruning*: We compute the probability of occurrence of each itemset in $\mathcal{F}$ in both the contact and non-contact databases. The probability of occurrence is simply the support of the itemset divided by the number of examples in the given dataset. For example, if itemset $X \in \mathcal{F}$, then the probability of its occurrence in $\mathcal{D}_C$ is given as $P(X, \mathcal{D}_C) = \sigma(X, \mathcal{D}_C)/|\mathcal{D}_C|$.

   As a first step in pruning we can remove all itemsets $X \in \mathcal{F}$ which have a greater probability of occurrence in the non-contact database than in the contact database, i.e., if $P(X, \mathcal{D}_{NC}) > P(X, \mathcal{D}_C)$. Actually, we compute the ratio of the contact probability versus the non-contact probability for X, and prune it if this ratio is less than some suitably chosen threshold ρ, i.e., we prune X if $P(X, \mathcal{D}_C)/P(X, \mathcal{D}_{NC}) < \rho$. In other words we want to retain only those itemset that have a much greater chance of predicting a contact rather than a non-contact.

5.2. Testing on Unknown Examples

The goal of the testing phase is to find how accurately the mined set of rules predict the contacts versus the non-contacts in new examples not used for training. We used a random 10% of the files in the database for testing. The test set had a total of 2,336,548 pairs, out of which 35,987 or 1.54% were contacts. Since we do know the true class of each example it is easy for us to find out how well our rules are for prediction.

For testing we generate a combined database $\mathcal{D}_t$ containing all pairs of amino acids in contact or otherwise. For each example we know the true class. We assign each example a predicted class using the following steps:

1. *Evidence Calculation* For each example E in the test dataset $\mathcal{D}_t$, we compute which itemsets in the set of mined and pruned closed frequent

itemsets $\mathcal{F}$ are subsets of E. Let's denote the set of these itemsets as S. We next calculate the cumulative contact and non-contact support for example E, i.e., the sum of the supports of all itemsets in S in the contact and non-contact database. Finally, we compute the evidence for E being a contact, i.e., we take the ratio of the cumulative contact support over cumulative non-contact support, denoted as ρ_E. Any example E with zero contact support is taken to be a non-contact and discarded, and only the examples or test pairs with positive contact support are retained for the next step.

2 *Prediction* To make the final prediction if a test pair of residues is in contact or not, we sort all test examples E (with positive cumulative contact support) in decreasing order of contact evidence ρ_E. Finally, the top γ fraction of examples in terms of ρ_E are predicted to be contacts and the remaining $1 - \gamma$ fraction of examples as non-contacts. How γ is chosen will be explained below.

5.3. Model Accuracy and Coverage

In predicting contacts versus non-contacts for the test examples, we have to evaluate the mined model based on two metrics: *Accuracy* and *Coverage*. Furthermore, we are only interested in the prediction of contacts; thus accuracy and coverage is only considered for contacts. Accuracy is the ratio of correct contacts to the predicted contacts, while coverage is the percentage of all contacts correctly predicted. Thus, accuracy tells us how good the model is, while coverage tells us the number of contacts predicted.

More formally, let N_{tc} denote the number of true contacts in the test examples, N_{pc} the number of predicted contacts, N_{tpc} the number of true predicted contacts, and let N_a denote the number of all possible contacts, i.e., $N_a = (N - 3) \times (N - 2)/2$ (where N is the protein length), since the contact map is symmetric and pairs with sequence separation less than 4 are ignored. The accuracy of the model is given as:

$$A = N_{tpc}/N_{pc}$$

The coverage of the model is given as:

$$C = N_{tpc}/N_{tc}$$

The number of contacts predicted N_{pc} of course depends on how we chose γ, since the top γ fraction of test examples based on evidence is predicted as contacts. Since a protein is characterized by N_{tc} true contacts, we set $\gamma = N_{tc}^*/N_a^*$ and then predict the top γ fraction of examples as contacts. Note that N_{tc}^* and N_a^* denote the actual contacts and all pairs, respectively, that have positive contact support, since we discard examples with zero contact support. By adopting the above method, the number of predicted contacts is limited to those actually present in the protein. Further, this method has been used by previous approaches to contact map prediction [FC99, OV97], and we retain it to facilitate comparison with previous results.

We also compare our model against a random predictor. The accuracy of random prediction of contacts is defined as:

$$A_r = N_{tc}/N_a$$

6. Experimental Results

We mined the pairs database using various combinations of context information and then tested the model on the unseen proteins. The pairs databases for training and testing had the following approximate sizes: $\mathcal{D}_C = 32MB$ for the training contacts database, $\mathcal{D}_{NC} = 2GB$ for the training non-contacts database, and $\mathcal{D}_t = 340MB$ for the testing database (includes both contacts and non-contacts). For all experiments below, we used a minimum support of 0.5% in the contact database, and we pruned a pattern if the ratio of contact to non-contact frequency was less than 4 (except for the amino-acids-only case, where we used a ratio of 1.5).

Amino Acids Only. Our first goal was to test how much information is contained in the amino acids only, i.e., for both training and testing, each example consisted of only the two amino acids a_i and a_j, and nothing else. Figure 3 shows the accuracy, coverage, and improvement of the mined model over the random predictor (legend "ratio" in the graph) for the test set. The improvement is the ratio of the accuracy of our model with respect to the accuracy of the random predictor. This ratio is one of the metrics used in past work on contact maps [FC99, OV97, TCS96]. The accuracy and coverage is the mean value over all proteins. The figure shows that the amino acids have some information that can be used to predict contacts versus non-contacts, but this information is not too good. The figure plots the accuracy and coverage as percentages. It also plots the improvement of the model over the random predictor. The x-axis shows the *prediction factor*, which is related to the γ value (used to predict the top fraction of pairs as contacts). The prediction factor is in multiples of N^*_{tc}, the number of true contacts in the protein with positive contact evidence. For example, a value of 10 means that the top $(10 \times N^*_{tc})/N^*_a$ fraction of the examples are predicted as contacts.

The top-most graph in Figure 3 shows the accuracy and coverage of the predictor over test proteins of all lengths. The other two figures below show how accuracy and coverage change with protein length. We have divided the

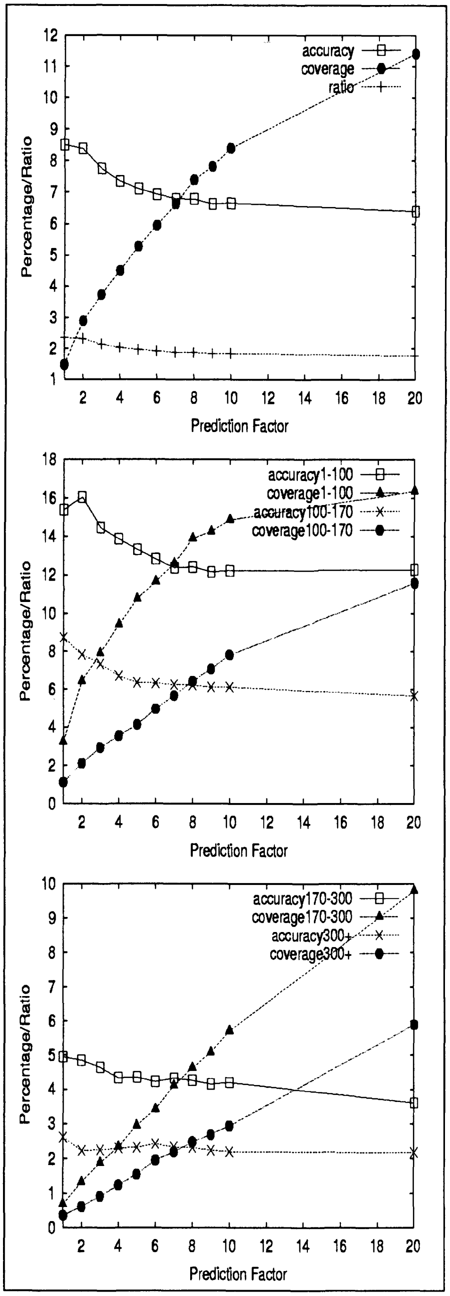


Figure 3. Amino Acids Only

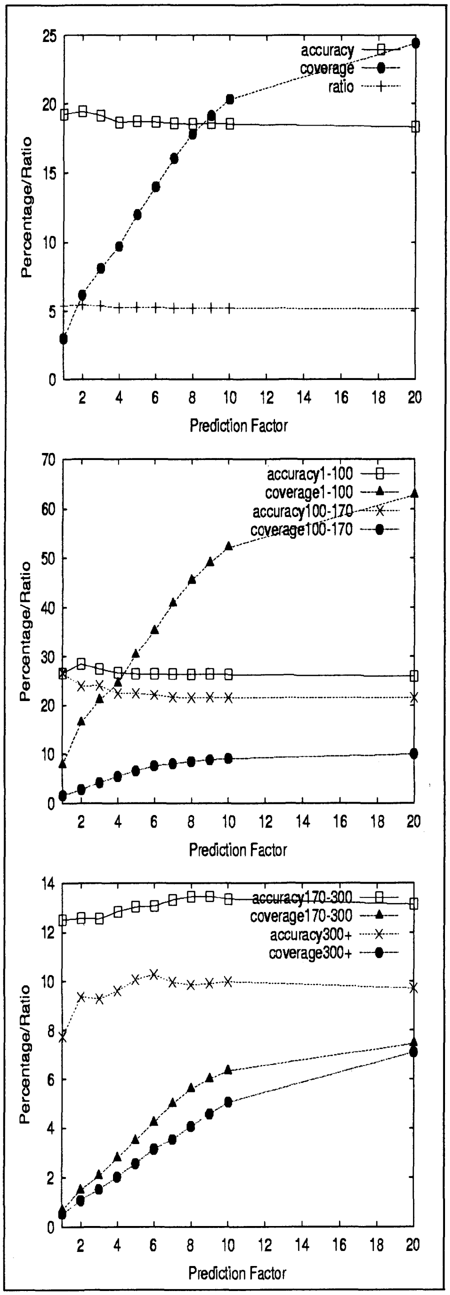
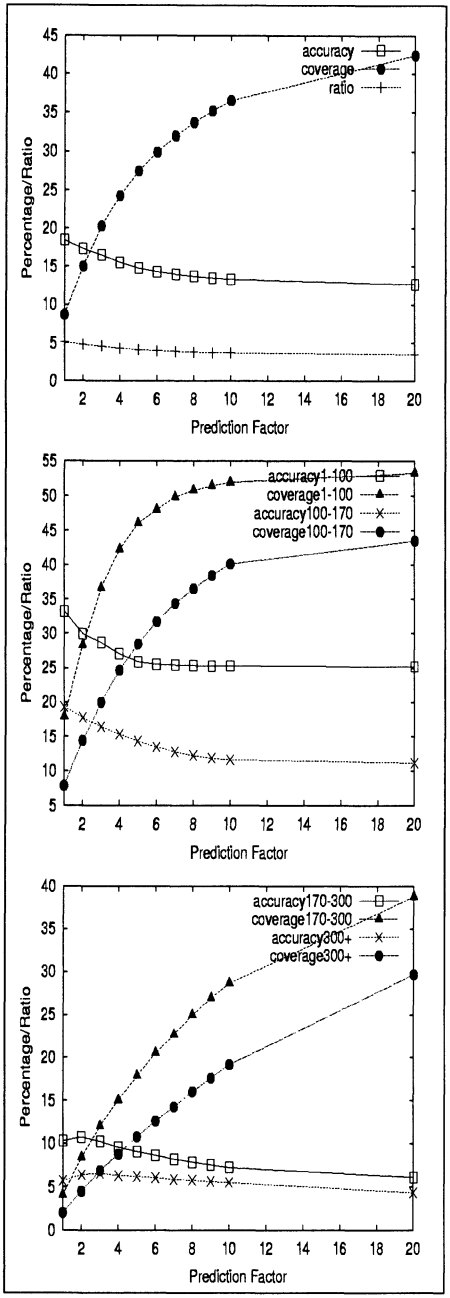


Figure 4. HMMSTR States and Amino Acids



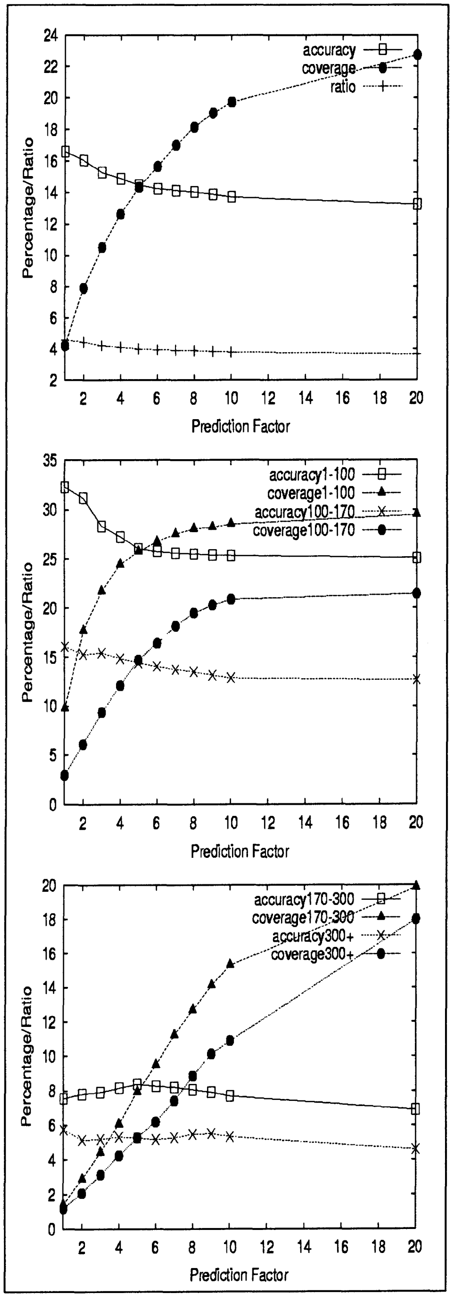


Figure 5. HMMSTR States, Amino acids, and R_t, D_t, C_t Symbols

Figure 6. HMMSTR States, Amino Acids, and Amino Acid Profiles

test proteins into four bins: $1 \leq N < 100$, $100 \leq N < 170$, $170 \leq N < 300$, and $300 \leq N$.

We find that over all proteins the amino acids by themselves can be used to give an 8.5% accuracy, 1.5% coverage, and an improvement over a random predictor by a factor of 2.4. Note also the interesting trend in the graph. As the prediction factor increases we get better and better coverage, but the accuracy trails off. This represents the classic accuracy versus coverage trade-off common to many prediction problems. Which value to choose for the prediction factor depends on what is more important. It has been reported in [VKD97] that the 3D structure of proteins can be recovered quite robustly, even from corrupted contacts maps. This implies that coverage should have an higher weight than accuracy. In any case, if we had to choose a value representing the best trade-off, we can pick the point where the accuracy and coverage curves intersect. This happens for a prediction factor of 7, where we have roughly 7% accuracy and coverage, and which is 2 times better than random. For 6.3% accuracy we can increase coverage to 14%.

When we consider the results for proteins of different lengths, we find the same trade-off between accuracy and coverage. Looking at the crossover point, we get around 13% accuracy and coverage for short proteins with $N < 100$, 6% for $100 \leq N < 170$, 4.5% for $170 \leq N < 300$, and around 2% of longer proteins.

HMMSTR States and Amino Acids. We next added the HMMSTR states corresponding to a_i and a_j, i.e., we added the columns $q_{i_1}, q_{i_2}, \cdots$ and $q_{j_1}, q_{j_2}, \cdots$ to the training and testing sets. Figure 4 shows the results. If we look at the cross-over point we get almost 19% accuracy and coverage, while the model remains 5.2 times better than random. For 18% accuracy we can get coverage of 25% (still 5.1 times better than random). Figure 7 shows the results in a slightly different format. It plots the improvement in coverage/accuracy over a random prediction. These results are comparable to or better than the results recently reported in [ZK00], where they examined pairwise amino acid interactions in the context of secondary structural environment (helix, strand, and coil), and used the environment dependent contact energies for contact prediction experiments. For about 25% coverage our model does more than 5 times the random predictor, as compared to the 4 times improvement reported in [ZK00]. Figure 8 shows the predicted contact map for the protein $2igd$ that we used in the introduction. We got 35% accuracy and 37% coverage for this protein. The figure shows the true contacts, the contacts correctly predicted, and all the contacts predicted (correctly or incorrectly).

If we look at proteins of various lengths in Figure 4, we find that for $N < 100$, we get 26% accuracy and 63% coverage at the extreme point (4 times over random). For $100 \leq N < 170$ we get 21.5% accuracy and 10% coverage to-

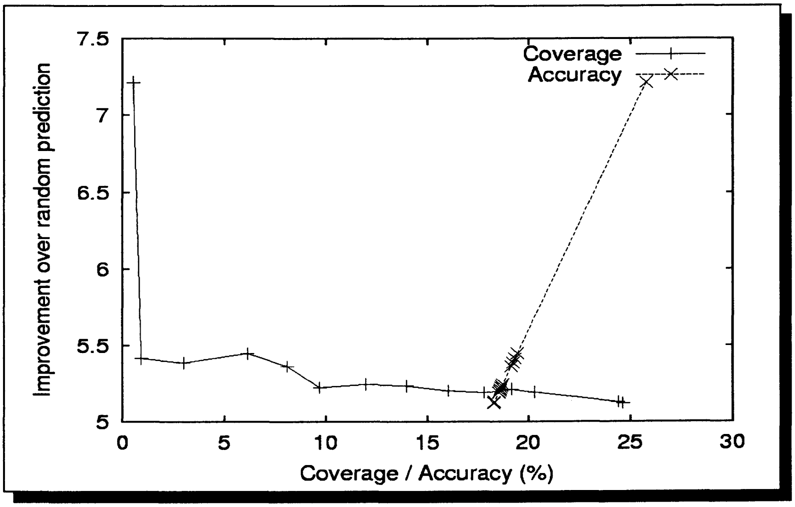


Figure 7. Improvement over Random Prediction

wards the end (6 times over random), for $170 \leq N < 300$ we get 13% accuracy and around 7.5% coverage (6.5 times over random), and for longer proteins we get 9.7% accuracy and 7.5% coverage (7.8 times over random). We believe these results are the best, or at least comparable to those reported so far in the literature on contact map prediction [FC99, OV97]. For example, Fariselli and Casadio [FC99], used a Neural Network based approach over pairs database, with other contextual information like sequence context windows, amino acid profiles, and hydrophobicity values. They reported an 14.4% accuracy over all proteins, with an 5.4 times improvement over random. They also got 18% accuracy for short proteins with an 3.1 times improvement over random. Olmea and Valencia [OV97] on the other hand used correlated mutations in multiple sequence alignments for contact map prediction. They added other information like sequence conservation, alignment stability, contact occupancy, etc. to improve the accuracy. They reported 26% accuracy for short proteins, but they did not report the result for all proteins. While we believe that our hybrid approach does better, we should say that direct comparison is not possible, since previous works used a different (and smaller) PDB_select database for training and testing. One draw back of these previous approaches is that they do not report any coverage values, so it is not clear what percentage of contacts are correctly predicted. Another approach to contact map prediction was presented in [TCS96], which was based on correlated mutations. They obtained an accuracy of 13% or 5 times better than random.

Adding Additional Information. We next tried to add more columns to the training database. For example we separately added the amino

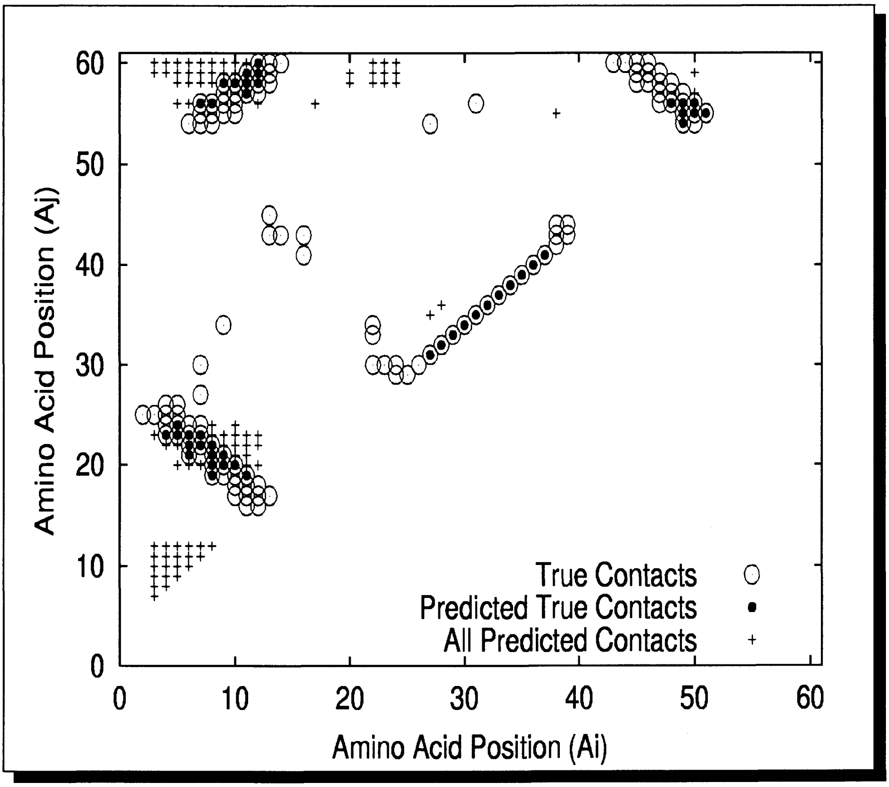


Figure 8. Predicted Contact Map (PDB file 2igd, $N = 61$)

acid profiles, and the structural context symbols for the 3-state secondary structure D_t, discrete backbone angle region R_t, and the context symbol C_t. The results for these cases are shown in Figure 5 and Figure 6. As we can see adding the profiles did not add any additional prediction power to our model, while adding the structural symbols had a positive (somewhat mixed) effect on accuracy and coverage. It appears that while the accuracy of the prediction drops a little there is tremendous boost in the coverage of the model. For example at around 18% accuracy we get about 25% coverage using the HMM states and amino acids (see Figure 4), but when we add the structural symbols, we get about 44% coverage for an accuracy of 12.5%. What this tells us is that the structural symbols can be helpful in providing the right context for the predictions and thus help in identifying a larger portion of the contacts.

In conclusion we have presented a new hybrid HMM and association rule mining method for contact predictions. Our results are the best or comparable to those previously reported. We are currently working to further improve both accuracy and coverage by carefully selecting many of the threshold parameters used in the experiments, as well as by adding additional attributes that might help prediction. Ultimately, the test of this method alongside all others will take place in the next round of Critical Assessment of protein Structure Prediction (CASP) [MPJF95], a world-wide blind prediction experiment, held every two years.

References

[AMS⁺97] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17), 3389-402, 1997.

[BB98] C. Bystroff and D. Baker. Prediction of local structure in proteins using a library of sequence- structure motifs. *Journal of Molecular Biology*, 281(3), 565-77, 1998.

[BTB00] C. Bystroff, V. Thorsson, and D. Baker. HMMSTR: A hidden markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, (to appear), 2000.

[Edd98] S. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9), 755-63, 1998.

[FC99] P. Fariselli and R. Casadio. A neural network based predictor of residue contacts in proteins. *Protein Engineering*, 12(1), 15-21, 1999.

[HB96] K. Han and D. Baker. Global properties of the mapping between local amino acid sequence and local structure in proteins. *Proc. Natl. Acad. Sci. U S A*, 93(12), 5814-5818, 1996.

[Hon99] B. Honig. Protein folding: from the levinthal paradox to structure prediction. *Journal of Molecular Biology*, 293(2), 283-93, 1999.

[HS94] U. Hobohm and C. Sander. Enlarged representative set of protein structures. *Protein Science*, 3(3), 522-524, 1994.

[KS83] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22, 2577-2637, 1983.

[MPJF95] J. Moult, J.T. Pedersen, R. Judson, and K. Fidelis. A large-scale experiment to assess protein structure prediction methods. *Proteins*, 23(3), ii-v, 1995.

[OV97] O. Olmea and A. Valencia. Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Folding & Design*, 2, S25-S32, June 1997.

[Rab89] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257-86, 1989.

[SMF92] L. Serrano, A. Matouschek, and A.R. Fersht. The folding of an enzyme. III. Structure of the transition state for unfolding of barnase analysed by a protein engineering procedure. *Journal of Molecular Biology*, 224(3), 805-18, 1992.

[TCS96] D. Thomas, G. Casari, and C. Sander. The prediction of protein contacts from multiple sequence aligments. *Protein Engineering*, 9(11):941-48, 1996.

[VKD97] M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. *Folding & Design*, 2(5), 295-306, September 1997.

[WF96] J. Wootton and S. Federhen. Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.*, 266, 554-71, 1996.

[WGK00] Y. I. Wolf, N. V. Grishin, and E. V. Koonin. Estimating the number of protein folds and families from complete genome data. *Journal of Molecular Biology*, 299(4), 897-905, 2000.

[YBR$^+$98] Q. Yi, C. Bystroff, P. Rajagopal, R. E. Klevit, and D. Baker. Prediction and structural characterization of an independently folding substructure in the src sh3 domain. *Journal of Molecular Biology*, 283(1), 293-300, 1998.

[ZK00] C. Zhao and S.-H. Kim. Environment-dependent residue contact energies for proteins. *Proc. Natl. Acad. Sci. USA*, 97(6), 2550-5, 2000.

Chapter 10

KDD SERVICES AT THE GODDARD EARTH SCIENCES DISTRIBUTED ACTIVE ARCHIVE CENTER

Christopher Lynnes and Robert Mack

Abstract NASA's Goddard Earth Sciences Distributed Active Archive Center (GES DAAC) processes, stores and distributes earth science data from a variety of remote sensing satellites. End users of the data range from instrument scientists to global change and climate researchers to federal agencies and foreign governments. Many of these users apply Knowledge Discovery from Databases (KDD) techniques to large volumes of data (on the order of a terabyte) received from the GES DAAC. However, rapid advances in computer power are enabling increases in data processing that are outpacing tape drive performance and network capacity. As a result, the proportion of data that can be distributed to users continues to decrease. As mitigation, we are migrating more knowledge extraction (*e.g.,* data mining and data reduction) activities into the data center in order to reduce the data volume that needs to be distributed and to offer the users a more useful and manageable product. This migration of activities faces several technical and human-factor challenges. As data reduction and mining algorithms are often quite specific to the user's research needs, the user's algorithm must be integrated virtually unchanged into the archive environment. Also, the archive itself is busy with everyday data archive and distribution activities and cannot be dedicated to, or even impacted by, the mining activities. Therefore, we schedule KDD "campaigns", during which we schedule a wholesale retrieval of specific data products, offering users the opportunity to extract information from the data being retrieved during the campaign.

Keywords: KDD Services, data center

1. Introduction

NASA's Goddard Earth Sciences Distributed Active Archive Center (GES DAAC) is responsible for processing, storing and distributing data from a variety of remote sensing satellite missions to users. More than simply

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 165–181.
© 2001 *Kluwer Academic Publishers.*

distributing the data to users, though, the GES DAAC's mission is to maximize the investment benefit of the Earth Observing System program by providing data and services to help people fully realize the scientific and educational potential of global climate data. The datasets that the GES DAAC supplies to the user community span a range of disciplines from oceanic parameters to atmospheric and climatic datasets (Table 1). Most of these datasets are global in coverage, at resolutions down to 250 m, and span several years to decades. As a result, data volumes are large and growing rapidly.

Table 1. Datasets and coverage available from the Global Change Data Center

Datasets	**Parameters**	**Spatial Coverage**	**Time Coverage**
Assimilated Time Series	Model output: surface fluxes, atmospheric profiles, radiation quantities	Global	1980-1993
Advanced Very High Resolution Radiometer (AVHRR) Pathfinder	Vegetation	Global	1981-2000
Coastal Zone Color Scanner (CZCS), Sea-viewing Wide Field-of-View Sensor (SeaWiFS)	Ocean Color (chlorophyll)	Piecemeal global	1978-1986 (CZCS) 1997-current (SeaWiFS)
Total Ozone Mapping Spectrometer, Tiros Operational Vertical Sounder (TOVS) Pathfinder	Ozone, atmospheric profiles	Global	1978-1999
Upper Atmosphere Research Satellite (UARS)	Atmospheric composition	Global	1991-1998
Various In Situ and Satellite Precipitation	Precipitation	Variable	1920-present (varies with datasets)
Tropical Rainfall Measuring Mission (TRMM)	Precipitation et al.	35 N to 35 S	1998-present
Moderate-Resolution Imaging Spectroradiometer (MODIS)	Calibrated radiance, ocean color, clouds, atmospheric profiles	Global	2000-

The end users of this data range from instrument scientists to global change and climate researchers to federal agencies and foreign governments. Many of these users apply Knowledge Discovery from Databases (KDD) techniques to terabyte-scale volumes of GES DAAC data. KDD encompasses a range of activities, including data selection, data reduction and data mining, in order to extract knowledge or information from large-volume databases. In the GES DAAC context, our "database" actually consists of the science data itself, stored as discrete files in robotic tape libraries, plus indexing information (*i.e.*, metadata) stored in a traditional relational database management system.

Table 2 shows some of the KDD applications for various user communities. The community has voiced a desire to have online access to the data for KDD applications [BDS+99]. Advances in magnetic disks have made this more feasible than ever for moderate volumes; however, for large volumes (*e.g.*, > 20 TB), this is not yet affordable for most large data centers. Thus, these data centers currently rely on tape technologies, whose storage density is growing at only about 30% per year [Sch00]. As disk technology advances, the affordable online volume will certainly increase; however, data volumes are expected to grow at a similar pace. Indeed, they have grown by orders of magnitude over the last few years (Figure 1).

Table 2. KDD applications for various user communities

User Community	**Research Area**	**Data Usage Pattern**	**Example of KDD Application**
Discipline Scientists	Derive results within one discipline (*e.g.*, biological oceanography)	Large volumes of a few closely related parameters	Long-term trends for global change studies
Interdisciplinary Scientists	Use/fuse data from multiple disciplines (*e.g.*,effect of global change on infectious diseases)	Data for many parameters and datasets, often for one geographic location	Correlation of features in disparate datasets
Instrument Scientists	Develop instruments and processing algorithms	Data for one instrument over long time periods	Long-term trend analysis of instrument behavior

The causes (or enablers) of this growth include Moore's Law, which implies a doubling of processing power every 18 months, or a growth rate of about 60% per year. Also contributing to the growth is the launching of instruments viewing the earth at higher resolution and with more channels. The new channels enable the derivation of yet more geophysical parameters as the science community continues to develop new algorithms to take advantage of them. For example, in the 1980's, the satellite-borne Coastal Zone Color Scanner, with five channels, provided information on ocean pigment (color). The current state-of-the-art ocean color instrument, the 36-channel Moderate Resolution Imaging Spectrometer (MODIS), is providing estimates of not just ocean pigment, but also such parameters as chlorophyll *a* concentration, chlorophyll fluorescence and phycoeryithrin concentration. A comprehensive list of MODIS parameters, which also include atmospheric and land parameters, is available in the MTPE/EOS Data Products Handbook [WM97].

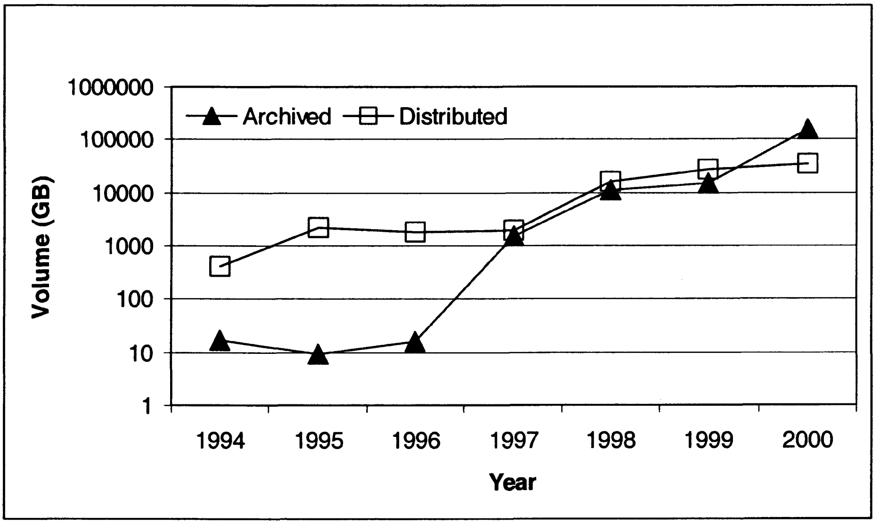


Figure 1. *Volume distributed and archived each year at the GES DAAC.*

While our ability to process data has been rising steeply, our ability to store and distribute data has not been rising so quickly, largely because advances in tape technology have been rising more slowly than advances in processors or even disk technology. Furthermore, this disparity is likely to grow over time. The areal density of magnetic tape media increases more slowly than the disk or processing speed; moreover, the speed of the macroscopic components of tape technology, such as stackers, robot arms and tape transports, increases even more slowly. One potential solution to the storage problem is to add more tape drives. However, tape libraries are typically limited in the number of drives that can be hosted, and adding drives usually sacrifices multiple tape slots. This limit does not apply to the distribution end, where a phalanx of standalone drives (usually in small tape stackers) can be supported if enough hosts and ports are available. However, each tape typically requires a few minutes of handling time (application of tape labels, loading and unloading, packaging, etc.) Ultimately, the cost of operations staff limits the volume that can be distributed through these means. Network distribution capacity is not subject to this limitation and has been increasing, but its capacity still lags behind tapes. Also, network distribution requires the user to have sufficient bandwidth and disk space to accommodate the data. Furthermore, network improvements are rapidly consumed by the enormous increase in general traffic on the network, a situation analogous to that of highway improvements attracting more development, in turn leading to renewed congestion.

The impact on data centers is clearly reflected in the three generations of the GES DAAC systems over the last eight years (Figure 1). The initial release of the system, Version 0, was fielded in 1994 and was designed to support ingest/archive rates of 2 GB/day and distribution rates of 40 GB/day, a ratio of 1:20. The Version 1 system, fielded in 1997, supports ingest of 75 GB/day and distribution of about 300 GB/day, a ratio of 1:4. The Version 2 system fielded in 1999, supports ingest of 400 GB/day and distribution of 400 GB/day, a ratio of 1:1. Subsequent hardware improvements are planned to increase both archive and distribution capacity, but always maintaining the 1:1 ratio. Furthermore, the distribution capacity is not designed to increase during reprocessing, at which time past data are processed using updated versions of the algorithm at rates of 2X. That is to say, data collected over two days are processed in one day. During reprocessing campaigns, only 1/3 of the volume entering the archive can be distributed, meaning that even if access patterns were uniform, most of the data would never reach users.

At the same time, user communities are beginning to express concerns about their ability to manage the data explosion at their end. Few researchers have the facilities to store and manage significant tape archives, let alone put several terabytes of data online. To mitigate this problem, many of them run all of the data through a data reduction algorithm immediately on receipt. Other users are taking advantage of remote data storage systems. An example is the San Diego Supercomputer Center's Storage Resource Broker [BMRW98] which lets users from universities and laboratories from around the country archive their data in the High Performance Storage Systems maintained at supercomputer centers.

While investments in data distribution technology can alleviate bottlenecks, it is difficult to see trends in technology that will completely solve the problem. Both data centers and users could benefit by minimizing the path between the source of the data and the knowledge extraction activity (Figure 2). One way to achieve this is to move the knowledge extraction algorithms to the data rather than the data to the algorithms. Clearly, significant savings can be achieved by applying the algorithms before distribution to tape or network; even more can be achieved by applying the algorithms within the processing stream.

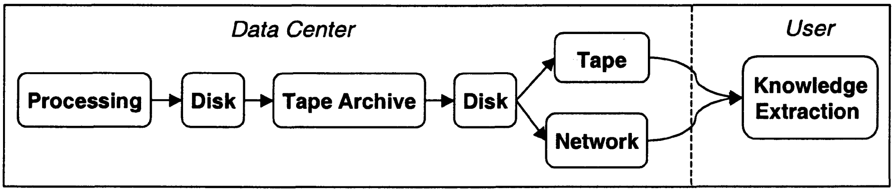


Figure 2. Data path from the initial source of data (processing) through distribution to the user's system, where the knowledge extraction process typically takes place.

Along these lines, the GES DAAC is establishing an architecture to support KDD at the data center. This effort faces a number of technical and non-technical challenges. Foremost among them is the need to support user-supplied algorithms. Some basic data reduction techniques can be of general use, such as spatial subsetting of remote sensing data over areas where field experiments are conducted (*e.g.*, the Southern Africa Regional Science Initiative, [JKDS96]). However, users have demonstrated a desire to control the algorithms used for knowledge extraction, as well as the file format of the output. Related to this is the need to adjust the algorithms (*e.g.*, through runtime parameters). Even if these hurdles can be overcome, there is sometimes a reluctance on users' part to work with data remotely.

On the technical side, there is some risk that the computing power required by the knowledge extraction algorithms exceeds that which can be provided by the data center. In this area, however, Moore's Law is finally in our favor, particularly with the advent of low-cost computing, such as personal computer clusters [BSS+95]. A more subtle challenge for data centers is the management of other resources, such as disk space and archive throughput: the data center must be able to execute the knowledge extraction without significantly degrading the distribution performance for the community at large. This last constraint virtually rules out KDD "on demand" at this time.

The key requirements, then, to support KDD within a data center are:

1. Retrieve input data from the archive in a fashion that minimizes the load on the archive, i.e., by minimizing the number of tape mounts.
2. Run user-supplied knowledge extraction algorithms on the input data, with minimal constraints on algorithm design or implementation.
3. Distribute the knowledge extraction products to the users (ideally making use of the existing distribution mechanisms within the data center).
4. Protect both KDD-support system and the data center against over-utilization of computer resources.

2. KDD Campaigns

In order to manage and minimize the impact of KDD activities on the data center, we conduct KDD "campaigns". In this context, a campaign is a pre-planned event with a defined duration during which we retrieve data from the archive in a controlled fashion. As the data are extracted from the archive, user-supplied knowledge extraction algorithms are executed and the results are shipped to the users. This concept of a campaign is analogous to reprocessing campaigns in which the data center feeds large amounts of data to data producers in order to support science processing. This scheme conveys two key advantages: (1) the data retrieval from tape can be optimized by asking for the full contents of a given tape at a time, and (2) the rate at which the KDD campaign proceeds can be adjusted daily or even hourly to fit within the demands of general distribution.

The campaign begins with the collection of the algorithms used for knowledge discovery activities. Currently, we support algorithms in C, Fortran and the commercial product IDL. Many of the received algorithms are written in IDL to take advantage of built-in functions for handling the Hierarchical Data Format (HDF) in which most of the GES DAAC's data are stored.

In addition to the supported languages, the algorithms must follow certain conventions:

Non-interactive/command-line. The programs must be executed from the command-line with no user interaction (*i.e.*, no prompting or Graphical User Interface). This allows the system to run automatically in off-hours without operator intervention.

Input/output filenames. The programs must accept input and output filenames from the command line. This places the data management responsibilities with the KDD-support system, rather than the knowledge extraction algorithm.

Empty data files. Some of the science algorithms which produce the input data intentionally generate empty data files when data are missing, rather than no data file at all. The information extraction algorithm must be able to recognize this and handle it gracefully.

Null-output KDD product files. Similarly, the knowledge extraction algorithms should generate files even when the output is essentially null, such as when no values are found to match content-based subset criteria. Thus, if the input data fail to generate useful output, the user will know whether the non-existence of output is due to the input data or due to a failure in the algorithm or KDD-support system. Alternatively, messages could be sent back to the user; however, the use of null-output product files is in keeping with the "batch" character of the KDD campaign.

Our initial step is to examine the code to verify that it will be able to run within the system. We then test the algorithms with small samples of data and benchmark them to determine how long they take to run, how much computing resources are used, and what data volumes are output. They are then integrated into the KDD-support system. After the algorithms are integrated into the system, an initial run is made on one day's worth of data and sent to the scientist for approval. Later, data covering one month are processed, and the results are again sent to the scientists for validation, with a notification that the algorithm will be activated for the campaign when a positive response is received from the scientist. This allows the scientist to verify that the results of the knowledge extraction algorithm are indeed the desired ones, and also permits some "tweaking" of the algorithm. At the same time, it mitigates against the risk of expending resources through an entire campaign, only to discover that the resulting output does not meet the user's needs.

The last phase is production, in which all of a given set of data products are retrieved from archive (usually in time order), and the algorithms are run to produce the knowledge extraction output, which is then shipped to the end user. While the integration and checkout phases are conducted by a specialist (interacting with the science users) in order to support debugging, this last phase is run by the operations staff. This allows them to properly adjust the rate of the KDD campaign within the load fluctuations resulting from general distribution. For example, they can reduce the data retrieval rate, and therefore the overall data flow rate through the system, at times when the normal operational load is high, such as when a number of large data requests are received from users. Likewise, they can increase the flow through the system during periods when the rest of the system load is light.

Control of the KDD process by GES DAAC operators is critical, due to the resource constraints of the current system. The DAAC is responsible for archiving data from flight missions, delivering data to science production facilities and shipping data to the user community, in descending order of priority. A lesson learned early in the KDD campaign was that the rate of the extraction activities, if high enough, can have an impact on some of these critical functions, such as the data archive. Also, the DAAC operators are able to retrieve the data in the most efficient manner possible, *i.e.*, by selecting batches of data that are known to be stored on the same tape. A more random selection process would increase the number of tape mounts required, drastically reducing the efficiency of both the DAAC and the KDD-support system. The disadvantage to DAAC control is that the scientists have little interaction with the process after the initial integration and test phase. Within the available resource envelope, it is possible for the scientists to be involved in setting the overall data flow rate. However, it is important to remember that all of the resources are shared: if more than one scientist is depending on output derived from a given input product, it is vital for each one to be prepared to receive the data at the operational rate so as not to impede the knowledge extraction on behalf of the other scientists.

3. KDD-support system Architecture

Overall, the KDD-support system is implemented as a system which is logically external to, but interfaces with, the GES DAAC system in a manner similar to that of other data producers (Figure 3). One key difference is a more direct interface to the archive which allows the data requests to be optimized to minimize tape mounts. After the data are received and processed by the KDD-support system, they are inserted back into the archive through an ingest interface similar to that used by external data providers. The KDD output data products are segregated from the standard data products available to the general user community. Distribution of the KDD output data to the scientist is accomplished using the GES DAAC's subscription and distribution subsystems. This mechanism thus supports all of the standard DAAC distribution mechanisms (network, 8mm and 4mm tape) and supports order tracking by the operations staff. The KDD output data are currently planned to be saved for a limited period of time (*e.g.*, 1 year) in case the scientist needs to recover them for one reason or another.

The KDD-support system itself is based on the Simple Scalable Script Based Science Processor (S4P). S4P is a set of Perl scripts developed at the GES DAAC to implement a generalized processing system. It is data-driven, following the paradigm of an assembly line: when a "work order" arrives at a "station", it is processed and the resulting output work orders are sent to downstream stations. Work orders are implemented as simple ASCII text files, while stations are implemented simply as Unix directories with configuration files describing the work to be done at that station. Each directory is monitored by a daemon process (called *stationmaster*) which detects the arrival of input work orders, forks off a job to process the work order, and then ships the output work orders to the downstream stations. It thus borrows heavily from the "services" concept of many operating systems, in which S4P services are invoked by the movement of files rather than connections to a port. Alternatively, an S4P implementation can be viewed as a straightforward implementation of a data flow diagram, in which each station implements a process and each work order implements a data flow. A simple Graphical User Interface has been developed to simplify the simultaneous monitoring of multiple stations and to support job management and troubleshooting.

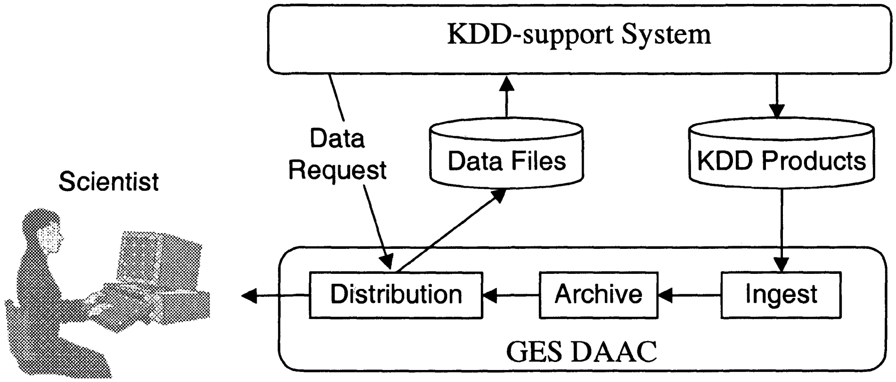


Figure 3. Interaction between the GES DAAC and the KDD-support system. The input data are obtained from the Distribution component of the GES DAAC and then processed through the KDD-support system. The output KDD products are ingested back into the GES DAAC for distribution to the End User.

Perhaps one of the most important features of the S4P system is that it places minimal requirements on the algorithms that are run within it, unlike many other processing systems which require use of a specific toolkit to interface with the data. This allows the acceptance of almost any algorithm, so long as it can be run from the command line. The only algorithms that cannot be accommodated are those which require interactive user input (*e.g.*, GUI-based tools).

The knowledge extraction algorithm is run within an ALGORITHM "station" under control of a Perl script (Figure 4). The rest of the stations in the KDD-support system control the data flow through the system:

- SPIGOT and QUEUE regulate the rate of the data flow.
- IMPORT, WATCHER and INGEST combine to acquire the data from the archive.
- TRIGGER determines which algorithms are ready to run.
- EXPORT sends output products back to the GES DAAC for archive and distribution.
- TRACK DATA keeps track of which data are currently available.
- CLEANUP deletes input and output products from the KDD-support system when they are no longer needed.

The KDD process consists of the following sequence (Figure 4):

(1) File Selection Interface. The operator selects the input data, using a simple graphical File Selection Interface. Typically, only the data type and version, start date/time and stop date/time need be specified.

(2) SPIGOT station. The output of the File Selection Interface is a work order which is deposited in the SPIGOT station, which regulates the rate at which requests are submitted to the DAAC's archive. This regulation is

crucial to the overall resource management of the KDD-support system and its impact on the GES DAAC system. The SPIGOT station allocates disk space for incoming data and sends work orders to an Import station, which requests the data from the archive.

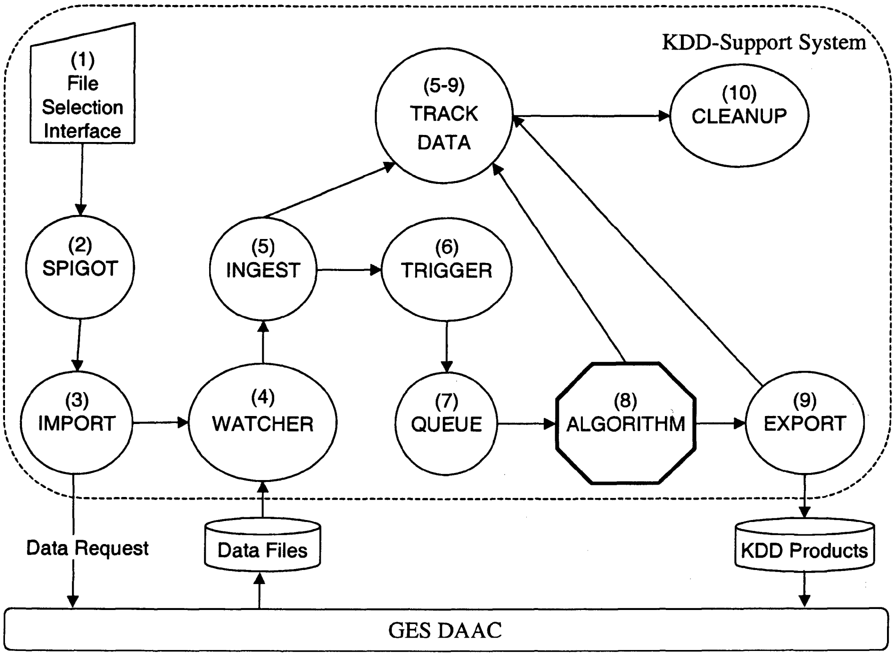

KDD-Support System
(1) File Selection Interface
(2) SPIGOT
(3) IMPORT
(4) WATCHER
(5) INGEST
(5-9) TRACK DATA
(6) TRIGGER
(7) QUEUE
(8) ALGORITHM
(9) EXPORT
(10) CLEANUP
Data Request
Data Files
KDD Products
GES DAAC

Figure 4. Architecture of KDD-support system. The "stations" are numbered according to the work flow sequence. The knowledge extraction activity takes place in the ALGORITHM station (hexagon), while the other stations manage the data flow.

(3) IMPORT station. This station makes the request to retrieve data from the archive.

(4) WATCHER station. In this station, a script polls a directory for the arrival of data, so that data processing can begin as soon as the input data are available to the KDD-support system.

(5) INGEST station. Here a script extracts key metadata and sends them to the TRACK DATA station (involved in steps 5 through 9) for storage in the Data Tracking database. The Data Tracking database tracks which data are currently available and whether a given input data granule has been run through all of the algorithms for which it is needed.

(6) TRIGGER station. The TRIGGER station resolves the data/algorithm dependencies and prepares the command-line to be executed, building unique output filenames from the input filenames. The output work order now has all of the information needed to run except the output location.

(7) QUEUE station. This station allocates disk space and specifies the output directory, storing information about disk space availability in a simple flat file database. While the other stations are "multithreaded" (i.e., they can have multiple jobs running simultaneously), the QUEUE station is singlethreaded to avoid database locking requirements.

(8) ALGORITHM station. This actually runs the knowledge extraction algorithm, sending output work orders to both the TRACK DATA station (indicating that the input data have been used) and the EXPORT station.

(9) EXPORT station. The EXPORT station compresses products with standard lossless compression (Lempel-Ziv-Welch) and inserts them into the GES DAAC system. From there the GES DAAC's normal distribution mechanisms are used to transmit the data to the scientist.

(10) CLEANUP station. Finally the CLEANUP station deletes input data that are "used up", *i.e.*, have had all the necessary algorithms run on them.

The initial campaign was begun on Tropical Rainfall Measuring Mission (TRMM) data with six algorithms from several investigators (Table 3), executing operations ranging from content-based subsetting to an algorithm which detects fires from radiance values [GKT00]. The reduction factors range from a factor of 4 to a factor of 37.

Table 3. Algorithms, TRMM input data and output data for the first KDD campaign.

Algorithm	Input Data	Output	Volume Reduction
Fire Detection	Visible-Infrared Radiance	Fire locations	14:1
Coincident subsetting	Precipitation radar rainfall	Satellite rainfall subsets coincident with rain gage locations	37:1
Content-based subsetting (4 algorithms)	Microwave, precipitation radar, and multi-instrument rainfall products	Subsets selected based on histograms and precipitation rates	4:1

The knowledge extraction algorithms are run on a 10-processor SGI Origin 2000, with 70 GB of disk space allocated to the activity. While this might seem like more than enough computing power for these purposes, the same machine is also used to ingest, archive and distribute TRMM data to the user community. As a result, we make an effort to keep the system utilization by the KDD activity at about 15%. Even with these self-imposed constraints, we are able to process data covering a week through the six knowledge extraction algorithms in less than eight hours. However, noticeable slowdowns in the KDD process occur when the GES DAAC is under load from major ingest, archive and distribution activities. Conversely, the ingest, archive and distribution activities of the GES DAAC are affected by KDD activities, though not enough to seriously impair mission-essential operations.

4. Future Directions

4.1. Performance Improvements

The most significant limiting factor for the kind of KDD campaign described above is contention with the data system's general distribution activities for access to the main data system archive. Therefore, if the data could be processed without having to make a special request of the archive, performance could be increased significantly. There are several potential scenarios to avoid retrieving from the archive. KDD on Ingest would execute the algorithms just after ingest (*i.e.,* extraction of metadata). It need not precede the archive step, as the two could proceed in parallel so long as the data were kept on disk until the KDD process was completed. This would require the algorithms be ready at the time of data creation. A closely related idea is to apply knowledge extraction algorithms to the data when reprocessed data are ingested. Reprocessing campaigns are common in NASA missions, typically occurring every one to three years as improvements are made to instrument calibration or to algorithms used to generate standard products. KDD on Reprocessing would allow KDD scientists to revise and improve their own knowledge extraction algorithms, running the improved version in the next reprocessing campaign. One drawback that should be noted is that low-level, or raw, data are typically not reprocessed. However, they may be extracted to drive the reprocessing of higher level (more processed) data, so that the algorithms could be applied to lower level data on distribution. Related to this concept is opportunistic KDD, that is, extracting information from files when they are retrieved for general distribution. Of course, some files may never be requested, so some effort would have to be made to fill in such holes. Also, the sequence could not be controlled, so that algorithms that need to span sequential data times could generally not be run in this mode. Finally, the data could be accessed for KDD when migrating the data to a new tape technology or even a data center, an activity which typically happens on a three- to five-year timescale. While this would provide access to the whole archive, migrations are rare activities, driven by other considerations. Also, some migrations are obviated by simply directing the reprocessing stream into the new device or data center. In the long run, data centers may have to offer varying levels of service, leaving it to users to evaluate the tradeoffs among performance, flexibility and even cost.

4.2. Algorithm Templates and APIs

The algorithm integration phase is currently manual and requires significant interaction with the algorithm provider. The provider also needs to know how to read the data formats and interact with the KDD-support system.

This phase could be shortened by providing templates in each supported language that simply read the data and write out a simple derived product of the data. A somewhat higher level of service could be provided by supplying an application program interface for each language that supported data access and subsetting capabilities. The Hierarchical Data Format-Earth Observing System (HDF-EOS) library already provides basic subsetting capabilities for HDF-EOS data, as does the Distributed Oceanographic Data Server (DODS) for HDF, netCDF and simple binary formats [GM95]. The Algorithm Development and Mining (ADaM) System for earth science applications at the University of Alabama-Huntsville goes even further, providing data access, preprocessing and even the mining algorithms themselves [RCGK00].

4.3. KDD-support Testbed/Sandbox

A complementary means of reducing the effort to integrate the algorithms would be to provide a testbed (or "sandbox") to the scientists, which would consist of an instantiation of the KDD-support software (perhaps on a separate machine) into which they could integrate their algorithms themselves. A substantial sample of data (say, data covering a month) would be available for testing, and the scientists would be given either accounts or a remote means of integrating and running the algorithms. This "do-it-yourself" integration would allow the data center to serve many more investigators and would save time by allowing them to see underneath the hood of the KDD-support system as they plug in their algorithms. However, it would clearly require a mature, robust KDD-support system.

4.4. Multiple Platform Support

While the KDD-support system currently runs only on SGI/Irix platforms, a prototype has also been tested for Linux. Both the S4P stationmaster and the scripts implementing the stations (*e.g.*, SPIGOT, IMPORT, WATCHER, etc.) in Figure 4 are written in Perl. This should make porting to other Unix-based platforms feasible, allowing incorporation of lower-cost platforms (such as PC's) as well as support of algorithms on their native platforms.

4.5. Transportable Applications and Agent-based KDD

As the KDD field matures, we expect many users, such as state and local governments, will want to employ knowledge extraction algorithms without having to develop the algorithms themselves. This raises the possibility of third-party providers of algorithms arising to fill the market need. Such a market is likely to be rapidly evolving. Eventually, these providers should be able to send the algorithms in real-time to the data center, along with the request for data; this would allow them to send the most up-to-date algorithm (Figure 5). The algorithm would be run within the data center against selected data, and the output of this KDD process would then be shipped back to the user.

A framework for transportable applications with associated data requests would be required for this, such as the MOCHA (Middleware Based On a Code Shipping Architecture) system [RR00]. In addition, the framework would need to accommodate resource limitations and chargebacks; much work is being done in this area to support E-commerce [MGM99]. Furthermore, such transportable applications may be agents which travel autonomously to a variety of data centers to develop the required results.

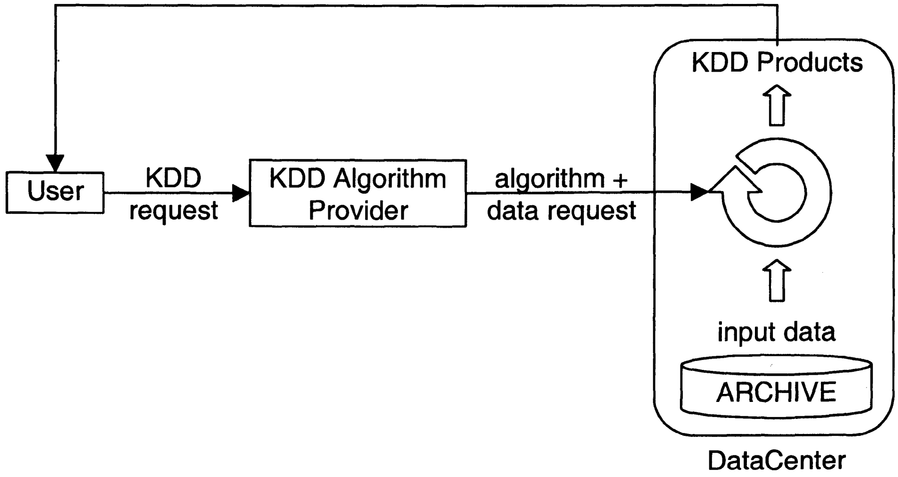


Figure 5. KDD services provided by transportable applications. The user requests information from a third-party provider of knowledge extraction algorithms. The provider sends a transportable algorithm to the Data Center, identifying desired data. The Data Center runs the algorithm and ships the output products back to the user.

5. Conclusion

It is difficult to forecast the effect of computer technology development over a ten-year or even five-year timeframe. While network and disk technology will undoubtedly advance, they are likely to be matched by processing advances, and thus data volume growth. Thus it is likely that data-center-based KDD services will be needed for some time to come in order to process the very largest datasets. The current growth of magnetic storage density is currently exceeding that of processing power, which may allow online storage of a higher percentage of the data. However, data access and throughput are not keeping pace with storage density; as a result systems supporting KDD for very large (multi-petabyte) archives are likely to require robust scheduling and resource management capabilities. If these challenges can be overcome, well-chosen KDD services at the data center should be able to offset rising distribution costs, while at the same time mitigating the data management problems faced by large-volume data users.

Acknowledgments

We gratefully acknowledge the funding provided by the Tropical Rainfall Measuring Mission project to implement the KDD system. We also thank Christian Kummerow of Colorado State University, Louis Giglio and Jacqueline Kendall of NASA/GSFC and Emmanouil N. Anagnostou of the University of Connecticut for providing KDD algorithms for the initial KDD campaign. We also thank Chandrika Kamath and an anonymous reviewer for many helpful comments on the manuscript.

References

[BDS+99] Behnke, J., E. Dobinson, S. Graves, T. Hinke, D. Nichols, P. Stolorz and P.Newsome, 1999. *Final Report on NASA Workshop on Issues in the Application of Data Mining to Scientific Data,* NASA/GSFC, 38 p.

[BMRW98] Baru, C., R. Moore, A. Rajasekar, M. Wan, 1998. The SDSC Storage Request Broker, Proc. CASCON '98 Conference, Nov. 30-Dec.3, 1998, Toronto, Canada, 12 p., *http://www.npaci.edu/DICE/Pubs/srb.ps.*

[BSS+95] Becker, D. J., T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawak, C. V. Packer, 1995. Beowulf: A Parallel Workstation for Scientific Computation, *Proceedings of the International Conference on Parallel Processing, http://www.beowulf.org/papers/ICPP95/icpp95.html.*

[GM95] Gallagher, J. and G. Milkowski, 1995. Data Transport within the Distributed Oceanographic Data System, *Fourth International World Wide Web Conference, http://www.w3.org/ Conferences/WWW4/Papers/67.*

[GKT00] Giglio, L., J. Kendall and C. J. Tucker, 2000. Remote sensing of fires with the TRMM VIRS, *International Journal of Remote Sensing*, **21**, 203-307.

[JKDS96] Justice, C. O., J.D. Kendall, P.R. Dowty and R.J. Scholes, 1996. Satellite remote sensing of fires during the SAFARI campaign using NOAA advanced very high resolution radiometer data, *J. Geophys. Res.*, **101**, 23851-23864.

[MGM99] Maes, P., R. H. Guttman and A. G. Moukas, 1999. Agents that buy and sell, *Communications of the ACM*, **42**, 81-91.

[RCGK00] Ramachandran, R., H. Conover, S. J. Graves, K. Keiser, "Algorithm Development and Mining (ADaM) System for Earth Science Applications," *Second Conference on Artificial Intelligence, 80th AMS Annual Meeting,* January, 2000, *http://ams.confex.com/ams/ annual2000/10satmet/program.htm.*

[RR00] Rodriguez, M., and N. Roussopoulos, 2000. MOCHA: a self-extensible database middleware system for distributed data sources, *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 213-224.

[Sch00] Schwarz, T., 2000. Magnetic tape as the mass storage medium, THIC Meeting for June 27 and 28, 2000, *http://www.thic.org/ Agenda_0600.html.*

[WM97] Wharton, S. and Myers, M. F. (ed.), 1997. MTPE EOS Data Products Handbook, Vol. 1, 270 p., *http://eospso.gsfc.nasa.gov/ftp _docs/DPH.pdf.*

Chapter 11

DATA MINING IN INTEGRATED DATA ACCESS AND DATA ANALYSIS SYSTEMS

Ruixin Yang, Menas Kafatos, Kwang-Su Yang, and X. Sean Wang

Abstract The rapid increase in the volume of scientific data sets has resulted in distributed data information systems applicable to Earth system science. Such a system should help users to locate data sets, to provide preliminary research results quickly and to support data deliveries under users' request. At George Mason University, we have been developing a data information system with both search and analysis components. In this system, three phases of data accesses are supported: phase one for meta-data search; phase two for on-line data analysis; and phase three for data ordering. For large volumes of data, searching on meta-data only will not be adequate. Scientists often need to search for data based on actual data values. This is a particular kind of data mining, which searches for data sets based on data content.

In this chapter, we first describe the system architecture. We then develop the concept of a data pyramid model and propose a histogram clustering technique for content-based searches. We use the model and the related technique to answer content-based queries approximately but efficiently. We will also describe our prototypes that integrate the content-based searches into a data information system.

Keywords: Data pyramid model, content-based search, histogram clustering

1. Introduction

In the last two decades, great progress has been made in Earth observing from space. Unprecedented observations of the Earth have provided very large amounts of data. The Earth Observing System(EOS) satellites *Terra* and *Aqua* and other Earth observing platforms are producing or will produce massive data products with rates of more than a terabyte

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 183–199.
© 2001 *Kluwer Academic Publishers.*

per day[AG99]. Moreover, with high performance computers, more and more model data are generated.

To use the rapidly growing volumes of data, one cannot rely on the traditional method where a user downloads data and uses local tools to study the data residing on a local storage system. Instead, users need to use data information systems, through which they can search for their interested data and can order the selected data sets or subsets. Since the Earth science data are naturally distributed among different data providers, the system should be distributed based on the existing data distribution.

CEOSR(Center for Earth Observing and Space Research) at George Mason University, together with GSFC(Goddard Space Flight Center) Earth Sciences Distributed Active Archive Center(GES DAAC) and the Center for Ocean-Land-Atmosphere Studies(COLA), have been developing a distributed data information system, named SIESIP(Seasonal to Interannual ESIP)[KWL$^+$98], which is a federated system and a part of a larger federation of Earth Science Information Partners(ESIPs)[NAS97]. SIESIP focuses on providing services to seasonal-to-interannual research communities as well as to interdisciplinary scientists and graduate students. Therefore, in addition to data searching and data ordering, the SIESIP system also provides services for data browsing and data analysis.

Data browsing and preliminary analysis are useful for interdisciplinary researchers to select data of interest to them. The usual data selection procedure is to specify a spatial/temporal range and to see what data sets are available under those conditions. Certainly, some textual restriction, such as keyword searches, could also be used to narrow down the data selection. As the data volume becomes larger and larger, users are not satisfied with locating data sets based on the regular textual/spatial/temporal conditions. Alternatively, users like to find data sets based on data values themselves. The histogram clustering technique may help users find the data sets efficiently with a specified accuracy so that it is used as a data mining tool for the data sets.

In the following, we briefly describe the overall SIESIP system architecture in section 2. In section 3, we emphasize on the on-line data browsing and data analysis components of SIESIP and give examples on using the system. We describe the data mining technique used for searching data value ranges in section 4. Finally, we conclude the paper and point out potential future work in section 5.

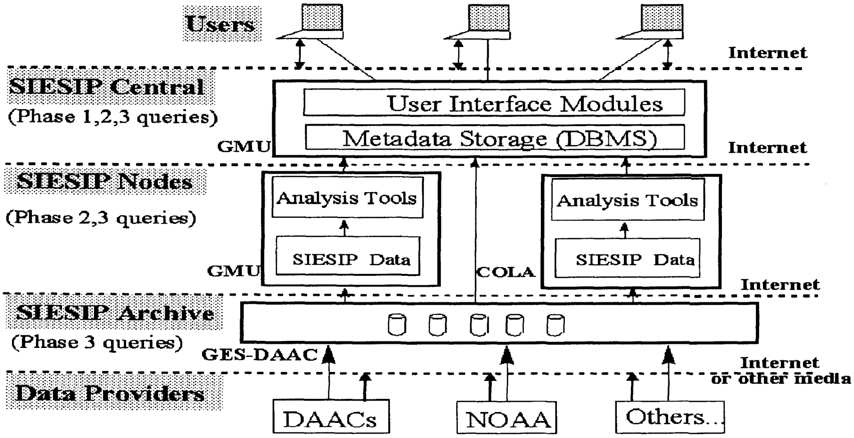

Users
SIESIP Central
(Phase 1,2,3 queries)
GMU
User Interface Modules
Metadata Storage (DBMS)
Internet
Internet
SIESIP Nodes
(Phase 2,3 queries)
GMU
Analysis Tools
SIESIP Data
COLA
Analysis Tools
SIESIP Data
Internet
SIESIP Archive
(Phase 3 queries)
GES-DAAC
Internet
or other media
Data Providers
DAACs
NOAA
Others..

Figure 1. SIESIP system sketch.

2. Design of SIESIP

SIESIP is one of the 12 science-focused ESIP-2 Federation projects selected by NASA[NAS97], now expanded to more than 15. We developed this data information system to serve data and other products to Earth scientists, especially to the seasonal-to-interannual(SI) researchers. Therefore, the system is science driven and is for serving science communities. Thus, the system has a "user-centric" emphasis based on different science needs as opposed to a "data-centric" emphasis followed in traditional (often government-run) data system design such as EOSDIS(Earth Observing System Data and Information System)[PKD$^+$94]. The system will provide not only data and information products but also data visualization, analysis and user support capabilities.

The SIESIP system has a client-server architecture (see Figure 1). The emphasis of the development efforts is on the server side to support various queries for information and data products. On the client side, some interfaces based on WWW(World Wide Web) technology are developed for a prototype purpose[YWK$^+$99]. Actually, the communication protocols to SIESIP servers are well defined so that users can develop their own client tools to access SIESIP servers. Three-level system access is currently supported. We termed it the three-phase data access model.

Phase 1 [Meta-data Access]: Using the meta-data provided by the SIESIP system, SIESIP users can find the data holdings. Meta-data knowledge is incorporated in the system, and users can issue queries to explore this knowledge.

Phase 2 [Data Discovery/On-line Data Analysis]: Users get a quick estimate of the type and quality of data found in Phase 1. Analytical tools are then applied as needed for users to mine content-based information, including statistical functions and visualization algorithms.

Phase 3 [Data Order]: After users locate the data sets of interest, they are now ready to order data sets. If the data are available through SIESIP, the information system will handle the data order; otherwise, an order will be issued to the appropriate data provider (e.g. GES DAAC) on behalf of users, or necessary information will be forwarded to users via e-mails for further action.

With the three-phase model, one can consider the server side of SIESIP as equivalent to various service and product providers with a series of servers. For prototype purposes, we also develop different kinds of clients to access the servers seeking for answers of queries of any phase. In terms of our implementation, we first develop components for special purposes, and then integrate several components into a system. The current SIESIP on-line data analysis system consists of components of data products, a database management system, communication protocols, data analysis tools, and user interface modules. These are designed for different functionalities and services.

Data product holding is one of the major components of the system. Presently, the system holds monthly mean data of Sea Surface Temperature(SST) and Normalized Difference Vegetation Index(NDVI) over global scales[KMea98], air temperature and precipitation covering South America[WM98], and other data sets such as special monthly TRMM(Tropical Rainfall Measuring Mission) data at different spatial resolutions. The raw data files in the system are used for analysis and ordering. The corresponding statistical summary data and holding catalogue are inserted into the DBMS(Database Management System).

A database management system is used for the system to handle catalogue meta-data and statistical summary data. Two major kinds of queries are supported by the database system. One is used to find right data files for analysis and ordering based on catalogue meta-data only. The other is queries on the data contents which are supported by the statistical summary data. For example, finding SST for given spatial and temporal ranges is a query belonging to the former group. However, a search for regions and time periods over which the values of mean SST are in a given value range is a query of the latter group.

To support on-line data analysis, analysis and visualization tools must be incorporated into the system. A free (public domain) software, GrADS (Grid Analysis and Display System)[DI95], developed by one of the SIESIP consortium members, COLA, is chosen for the system. The employment of the tool is fully automatic and transparent for the on-line analysis aimed at serving interdisciplinary science. Users actually do not know and do not need to know what tool the system is using other than the analysis functions used. With the current modular design, other tools can be easily integrated into the system.

The user interface modules are responsible for communications between SIESIP servers with SIESIP clients. On the server side, we use a Java-based "server" program to pick up communication messages at a specific port. Servlet programs are also developed for supporting messages through the HTTP(Hypertext Transfer Protocol) protocol. To test the system, we also developed a prototype GUI(Graphic User Interface) in Web environment. The main web page of the current SIESIP contains an applet based on Java Swing technology. A snapshot of the GUI is given in Figure 2. There are three menu lists for handling the three types of queries, respectively. The workplace menu list gives users control on choosing working data product objects for further analysis and other functions.

We use XML(eXtensible Markup Language) as the system language in SIESIP. Queries into and query results out of SIESIP servers are both in XML format. Information coded in XML could be easily used since various tools and software are available. SIESIP client developers could incorporate such tools and software into the client software. Since queries to all three phases are of uniform format, they are transparent to users. Therefore, we have a seamless integrated system to support all three phase queries by means of a system language based on XML.

3. On-line data browse and analysis

Since the SIESIP prototype GUI is a web based system, the only requirement to use the system is a web browser supporting Java and, of course, a network connection. The three phase queries[KWL$^+$98] are orthogonal to each other. Therefore, users can go to any queries directly. One procedure to use the system is described here. A user starts the system by visiting the SIESIP URL(Uniform Resource Locator). Instead of using the default settings, the user may use the "Workplace" menu list to find a specific parameter he/she is interested in. Then he/she can use "region" and "time" panels to restrict spatial and temporal coverage. Finally, the user can use the "Analysis" menu list to submit

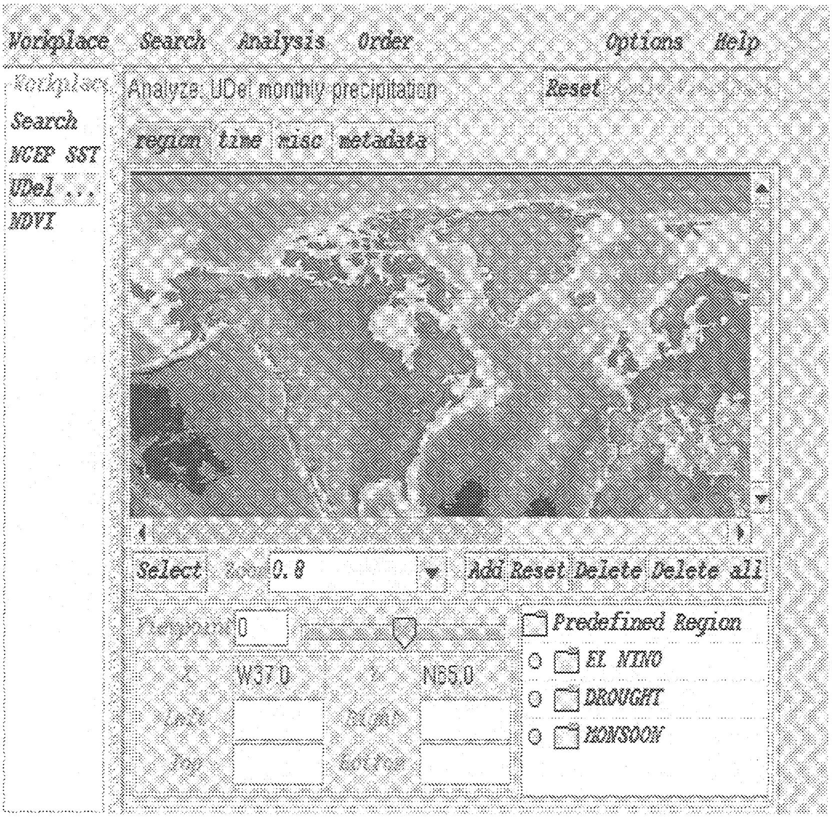

Workplace
Search
Analysis
Order
Options
Help
Workplace
Search
NCEP SST
UDel ...
NDVI
Analyze: UDel monthly precipitation
Reset
region
time
misc
metadata
Select
0.8
Add
Reset
Delete
Delete all
0
W37.0
N55.0
Predefined Region
EL NINO
DROUGHT
MONSOON

Figure 2. SIESIP GUI.

an analysis request, such as "Browse Image". After the server completes the requested job, it will send back the necessary information (an URL) to the client which will automatically pull another web page displaying the result. Based on the browse results, the user can then proceed with other on-line analysis functions or end this phase and proceed to order the data.

As stated above, SIESIP is one of the ESIP federation partners. The overall system is distributed and, therefore, requires data interoperability. The current SIESIP system supports data interoperability based on FTP(File Transfer Protocol) and DODS(Distributed Oceanographic Data System)[GM95] protocols. Some data sets in the SIESIP data set list are actually located in locations different from the main engine. If a user selects such a data set for analysis, the system will fetch the data through FTP or directly access the remote data by DODS protocol. No matter where the data sets are located or which protocol is used for remote data sets, the entire process is transparent to the system users. The on-line data analysis with data interoperabilities makes it very efficient for users to browse data sets.

In addition to the browse images, SIESIP also supports several other functionalities. One of them is the time series of geophysical variables. When users select a spatial area through the Java interface by "Rubber-Band" and give a temporal range, the system will compute the spatial average over the given region and return a time series. One step further, when users choose multiple variable/spatial region combinations, the system will return not only the time series displays but also scatter plots for all pairs with the corresponding correlation coefficients. Other kinds of correlations such as region-point, point-point temporal correlations (results are spatial distributed values) and spatial correlation (result is a time series) are also supported by the SIESIP system.

The browse images and the correlation analysis results generated on-the-fly can certainly be used for data mining purposes. For example, the region-point correlation function will first compute a time series which is the spatial average of a given geophysical variable. Then, the system will do the temporal correlation between this time series and the time series at every point in another given spatial region. The result is the spatial distribution of the temporal correlation coefficients. By just viewing the resultant image, users can quickly tell over which regions the studied variable has high temporal correlation with the other signal. The signal could be a spatial-averaged time series as described above. Or it could be a well-defined signal such as the Southern Oscillation Index(SOI) which is associated with the well-known El Niño and La Niña phenomena.

Data mining techniques help scientific data users not only in finding rules or relations among different data but also in finding the right data sets. With the explosively increasing data volume, users need a fast way to search for data of interest to them. In this process, users need to search data based on not only meta-data but also actual data values. For example, a user may be interested in the regions over which a given parameter has values in a certain range. A more detailed example related to current SIESIP on-line analysis functionalities is: to find an area where the temporal correlation coefficients between local NDVI(Normalized Difference Vegetation Index) anomaly values and SOI is larger than a certain value, say 0.5. To answer this kind of content-based (or value range) query, we need an efficient way to mine the data values. Under SIESIP, we develop a data pyramid model and create aggregation values to approximately answer the query. Details of the technique are discussed in the next section.

4. Data mining technique

The goal of our data mining is to find spatial regions and/or temporal ranges over which parameter values fall in certain ranges. The main challenge of the problem is the speed and the accuracy because they affect each other inversely. We try to balance the two factors, i.e., to rapidly answer value range queries with reasonable accuracies. Instead of finding *all* regions satisfying the conditions, we want to **quickly** find not less than a certain percentage (say, 95%) of the total qualified regions. This kind of approximate queries are termed statistical range queries [LWKY98, YYKW00].

4.1. Method description

Our method is based on a two level data pyramid model[YYKW00]. The problem domain which may be spatial areas or temporal ranges is divided into lower resolution cells. Each cell actually contains a number of original data values. One way to speed up the query processing is to pre-compute aggregations on all cells and use the aggregation values to approximately answer queries. The statistical summary data in the SIESIP DBMS are actually aggregation values, and spatial average is used to evaluate value ranges. Here we use histograms as the aggregation, and moreover, we go one further step to cluster the resultant histograms as well as the associated cells. The value range queries are answered on cluster basis instead of on cell basis. If the higher resolution (original resolution) result is needed, we can drill-down into individual cells, but only to cells selected already.

The major steps to build a data pyramid are: cell divisions, histogram creation, computing distances between histograms, clustering, and histogram summation. We will use a two-dimensional data set as an example to describe the above procedures as well as the query execution process.

Suppose that we have an Earth science data set which covers the globe for a certain time and of a specific spatial resolution. We can divide the problem domain, the globe, into disjointed cells. All cells together will just cover the Earth. The cell boundaries may be political boundaries, natural boundaries such as coastal lines and rivers, or artificially defined lines such as parallels and meridians. We need to make the cells large enough to include a significant number of data values to make the histograms meaningful. However, we should also limit the size of cells to capture local features.

On each cell, we create one histogram by binning the data values inside the cell. To get a histogram with high accuracy and a fixed number of bins, the best way is to use adaptive bin sizes according to the minimum and maximum values associated with this histogram. However, we use overall histogram similarity to cluster the cells. To compute similarities or distances between two histograms, we need to have all the histograms cover the same value range with the same bin sizes. Therefore, we choose the global minimum and maximum values to define an universal histogram frame.

Since we cluster the histograms and the associated cells, we need to measure the similarity or dissimilarity among the histograms. In this work, we simply compute the distances between all histogram pairs. We treat each histogram as a one-dimensional array, or a vector. The distance between two vectors can be measured with a specific metric. Here, the major metric used is the Manhattan metric[VR94] in which a distance between two vectors is measured by summing up the absolute difference of the corresponding components of the two vectors. Other metrics are also tested and the results based on them will be discussed towards the end of section 4.2.

After the distances between all pairs of histograms are computed, we group the histograms based on clustering methods. Although the clustering problem is unsupervised, new data values of incremental volume need to be classified based on the clusters generated. In this work, we only used the group-average, agglomerative hierarchical method and k-means method. Before the clustering result is used to support the value range queries, a decision must be made on selecting the number of clusters. There is no practical optimal method for picking a particular number of clusters. Therefore, we test several numbers of clusters in a given problem and pick a practical value.

The last step for building a data pyramid is to summarize the histograms in each cluster together to get a representative histogram for each cluster. At the same time, cells associated with each cluster are indexed in the same way. For example, suppose we choose six clusters based on the cluster analysis and our experience, we then obtain six representative histograms. All cells are identified with numbers 1 to 6. That is, each cell belongs to one and only one cluster.

When a statistical range query is issued, the system computes the count of data values in the particular range based on the representative histograms. The clusters are then ordered according to their counts. For any defined accuracy, a certain number of groups are included in the search. The accuracy is defined as the ratio of the selected number of data values to the total number of data values in the given value range. As more and more groups are included in the search, higher and higher accuracy is, obviously, reached. The advantage of this clustering method is that the clusters are ordered. Therefore, clusters of high contribution to the result are included first. Consequently, the accuracy increases quickly with the first few clusters. After the cells of high contribution to the query result are selected, the system can identify satisfactory data points in the original resolution by scanning the selected cells only. Therefore, the searching time is reduced by searching only a fraction of the total number of cells. The method guaranteed that more than certain percentage of qualified points are selected.

4.2. Experiments

We use a NDVI data set[KMea98] as an example to test the method. The NDVI values provide a strong vegetation signal and are used widely for classification of vegetation. In the classification process, finding areas for which NDVI values fall in a certain range is useful. So are the statistical range queries. The NDVI data we are using are monthly gridded with 1 × 1 degree spatial resolution in both longitude and latitude. The spatial coverage of the data set is global land. Since we divide only the spatial domain into cells, a fixed time, say August, 1981, was randomly chosen.

We divide the globe into 5 × 5 degree cells. For each cell, there are up to 25 associated NDVI values. We map those values into a fixed bin frame with 22 bins of width 0.05 covering the value range from -0.1 to 1.0. Since the NDVI data cover only global land, some cells are composed of less than 25 values because the corresponding cells cover both land and water. There are even many empty cells, i.e., cells associated with only invalid NDVI values. We just keep the resultant histograms with at least one valid value.

The histograms are clustered based on the methods described above. After we pick a cluster number, we sum up all histograms in each cluster into one representative histogram. Figure 3 shows those representative histograms for a case of six clusters. It is obvious that each cluster has a different dominant value range. This is the fundamental reason in speeding up the searching process to answer a statistical range query.

Suppose that we have a statistical range query: find cells, which, together, are associated with more than 95% of NDVI values between 0.56 and 0.78. Table 1 lists the related values for answering this specific query. First, the system computes the contribution from each cluster based on its representative histogram and orders them. In this case, the order is cluster 1, 3, 2, 5, 4, and 6. The order is associated with the count of values in the given range in descending order. The contribution from each cluster, that is, the percentage of values in the given range for

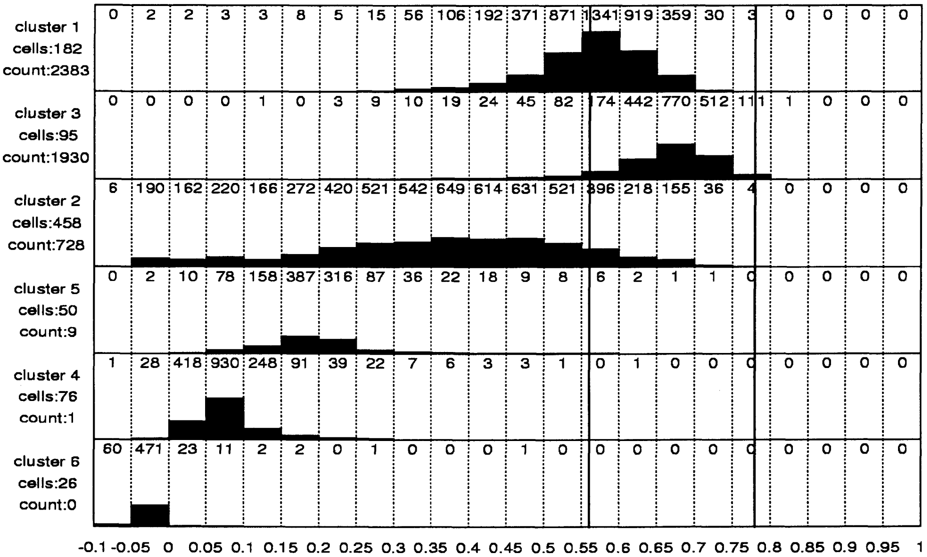


Figure 3. Representative histograms of six clusters.

Table 1. Query result for NDVI values between 0.56 and 0.78.

ID	cells	count	accuracy	saving 1	saving 2
1	182	47.35	47.35	92.98	79.48
3	95	37.99	85.35	89.31	68.77
2	458	14.44	99.79	71.64	17.14
5	50	0.19	99.98	69.71	11.5
4	76	0.02	100	66.78	2.93
6	26	0	100	65.78	0

the cluster, is summed up in the order, and the result is the accuracy. From Table 1, we can find that after including only 3 clusters, we have obtained an accuracy 99.79%. In other words, all cells in the first 3 clusters are associated with more than 99.79% NDVI values in the given range. By scanning on those selected cells, we are guaranteed to have more than 95% searched values.

How much time is saved by the clustering method? We measure the time saving by computing the ratio of the number of the selected cells

Table 2. Dependence on number of clusters

total	selected	accuracy	saving 1	saving 2
3	2	99.5	66.8	2.93
4	3	99.5	66.8	2.93
5	3	99.3	70.5	13.6
6	3	98.8	77.5	34.2
7	3	98.8	77.5	34.2
8	3	98.6	78.7	37.7
9	3	98.6	78.7	37.7
10	4	98.6	78.7	37.7

Table 3. Result for data without empty cells.

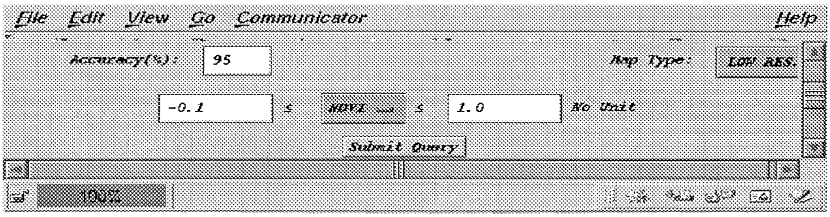
ID	cells	count	accuracy	saving 1	saving 2
4	481	36.86	36.86	81.44	81.44
5	290	35.93	72.79	70.25	70.25
1	541	26.94	99.73	49.38	49.38
2	318	0.23	99.95	37.11	37.11
3	670	0.05	100	11.27	11.27
6	292	0	100	0	0

to the total cell number. If we include the empty cells in the total, the time saving rate is 71.64% shown as "saving 1" in the Table 1. If we eliminate empty cells, the saving rate dropped to 17.14% as displayed in the last column. Note the time saving depends on many factors such as cluster number, data range, and data value distribution. Our study demonstrates that the effect of cluster number is very limited as long as the number is large enough, say, 6 or larger, as illustrated in Table 2. The value range and value distribution is problem dependent. For the same problem, the time saving rates for value range between 0.04 and 0.26 are 77.5% and 34.2%, respectively. To isolate the efficiency dependence on clustering method only, we used the method to a global data set covering both land and ocean, therefore, there were no empty cells. For one of the cases, we get more than 99% accuracy by including 3 clusters in total 6 clusters. The time saving is about 50% as shown in Table 3[YYKW00].

The effects of different distance measurements on the results are also studied. We compared the Manhattan metric to the Euclidean and maximum distance metrics from the case of NDVI with a 95% given accuracy

Table 4. Dependence on distance measurement methods

method	total cells	accuracy	saving 1	saving 2
Manhattan	584	98.8	77.5	34.2
Euclidean	855	97.5	67.0	3.61
Maximum	848	98.5	67.3	4.40



Accuracy(%): 95

Map Type: LOW RES.

-0.1 ≤ NDVI ≤ 1.0 No Unit

Figure 4. WWW mining interface.

and a value range between 0.04 and 0.26. Table 4 lists the results for the three metrics. The Manhattan metric gives the best results. The maximum distance method is simply taking the largest difference of all corresponding component differences between two vectors. The maximum metric should not give a very good result since it does not consider the shapes of histograms at all. The Euclidean metric is more sensitive to outliers. With a fixed number of clusters similar to that used with Manhattan metric, a few histograms with outliers will occupy a same cluster and force the majority of others to share the left-over cluster space. This results in one group to include most histograms, and of course, reduces the efficiency of the clustering method. Since the shape of histograms is the most important criterion for clustering, Q-type measurements[Job92] such as correlations would possibly lead to a better result. More studies are needed to confirm this.

4.3. Data mining in WWW environment

Our final goal in developing data analysis and data mining techniques is to make them accessible through the World Wide Web so that users can perform a similar mining without traditional tool and data downloading. Therefore, we have prototyped the clustering method in the WWW environment. The data pyramid is created on the server side as

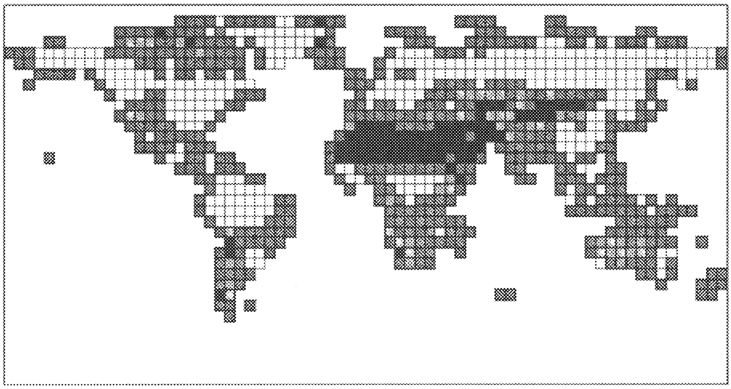


Figure 5. Map of selected cells and ignored cells.

pre-computing results. The interface shown in Figure 4 is used for Web users to make value range queries. An user should select a geophysical variable, a given value range, and a required accuracy. After submitting the query, the server will perform all procedures described before, and generate a color map indicating selected areas. Figure 5 is a similar map but in gray scale. The darkest cells are cells with the highest contribution to the query and so on. All shaded cells together satisfy the given accuracy.

5. Conclusions and future directions

The SIESIP system provides a Web interface for interdisciplinary scientists and graduate students to do preliminary data analysis with limited functionalities. The clustering method is also suitable for the same audiences under the same environment. Integrating the data mining method based on the clustering technique will make the SIESIP on-line analysis system more powerful. However, more work should be done for an operational system.

The first task is to add more data sets to the system. Data mining makes more sense when the data volume is huge (Gigabytes or more). What data sets should be included in the system is a science issue. We need to work with domain experts, Earth scientists, to achieve this goal. At present, SIESIP has a large volume of Earth science data. Putting these data sets on the SIESIP on-line analysis engine is simple. However, since pre-computing is required for building data pyramids for content-based queries, more studies are needed to find more effi-

Table 5. Comparison of CPU time required for hierarchical and k-means methods.

Data Set	Method	CPU(sec.)
Land (N=887)	k-means	0.46
Land (N=887)	average	3.62
Ocean (N=2005)	k-means	0.69
Ocean (N=2005)	average	15.78
Global (N=2592)	k-means	0.34
Global (N=2592)	average	25.35

cient strategies to create the data pyramids and to make SIESIP data available for content-based search. According to our tests, the most time consuming pre-computing is on the distance calculations between histogram pairs and the execution of the clustering algorithm. Very preliminarily, we tested the k-means clustering algorithm against the hierarchical method we used initially. The results for various numbers of histograms (N) on the same computer is shown in Table 5. For all three cases with data sets over land only, ocean only, and whole globe, k-means saves significant computing time. Actually, k-means needs only $O(N)$ operations, but the hierarchical algorithm needs $O(N^2 \log N)$ operations. Certainly, k-means is sensitive for initial setting. Different clustering methods[NH94, ZRL96] should be considered for treating large data sets. CLARANS(Clustering Large Applications based on RANdomized Search)[NH94] is based on randomized search, providing effective spatial data mining for both spatial attributes and non-spatial ones. BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies) [ZRL96] provides incremental and dynamic clustering method for multidimensional data points with memory and time constraints. BIRCH can be useful to cluster Earth science data sets with multiple time periods which usually have memory size problems to handle.

Another improvement on the current clustering technique is to include size condition on the query criterion. With the current cell division method, the cell size and shape are fixed. For scientific purpose, it is more useful to find areas larger than a specific size. For example, we may want to find areas on which NDVI values are less than 0.5 *and the area sizes are larger than* $1,000km^2$. The area locations are open and also the shapes of the areas are arbitrary. A potential solution is to use the technique described here to find high resolution points and then reconstruct areas based on the given conditions.

Acknowledgments

We acknowledge partial funding support from the NASA ESDIS ESTO prototype Project (NAG5-8607, NAG5-10177), from the Earth Science Enterprise WP-ESIP CAN Program (NCC 5-306), as well as from George Mason University. Wang also acknowledges partial support from NSF under the career award 9875114.

Other members of the SIESIP team include: B. Doty, J. Kinter, C. Steinmatz, & J. Wielgosz (COLA); L. Pham, & G. Serafino (GES DAAC); P. Chalermwat, L. Chiu, Z. Liu, J. McManus, D-B. Shin, Y. Vikhlyaev, J. Vongsaard, C. Wang, H. Weir, & Y. Xing (GMU); K. Miyakoda, P. Schopf & J. Shukla (GMU & COLA); H. Wolf (IRMA & GMU); and K. Matsuura & C. Willmott (UDel).

References

[AG99] G. Asrar and R. Greenstone, editors. *1999 EOS Reference Handbook.* NASA (Washington, DC), 1999.

[DI95] B. Doty and J. Kinter III. Geophysical Data Analysis and Visualization using GrADS. In E.P. Szuszczewicz and J.H. Bredekamp, editors, *Visualization Techniques in Space and Atmospheric Sciences*, pages 209–219. NASA, Washington, D.C., 1995.

[GM95] J. Gallagher and G. Milkowski. Data Transport within the Distributed Oceanographic Data System. In *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, pages 691–702, 1995.

[Job92] J. D. Jobson. *Applied Multivariate Data Analysis*, volume 2. Springer, 1992.

[KMea98] H. L. Kyle, J. M. McManus, and S. Ahmadand et al. *Climatology Interdisciplinary Data Collection, Volumes 1-4, Monthly Means for Climate Studies.* NASA Goddard DAAC Science Series, Earth Science Enterprise, National Aeronautics & Space Administration, NP-1998(06)-029-GSFC, 1998.

[KWL^{+}98] M. Kafatos, X. S. Wang, Z. Li, R. Yang, and D. Ziskin. Information Technology Implementation for a Distributed Data System Serving Earth Scientists: Seasonal to Interannual ESIP. In Maurizio Rafanelli and Matthias Jarke, editors, *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 210–215. IEEE, Computer Society, 1998.

[LWKY98] Z. Li, X. S. Wang, M. Kafatos, and R. Yang. A Pyramid Data Model for Supporting Content-based Browsing and Knowledge Discovery. In Maurizio Rafanelli and Matthias Jarke, editors, *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 170–179. IEEE, Computer Society, 1998.

[NAS97] NASA. NASA Selects Earth Science Information Partners. *NASA Press release, Dec. 2, 1997*, 1997. http://www.nasa..gov/releases/1997/.

[NH94] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proce. of the 20th VLDB Conference Santiago, Chile*, pages 144–155, 1994.

[PKD+94] R. D. Price, M. D. King, J. T. Dalton, K. S. Pedelty, P. E. Ardanuy, and M. K. Hobish. Earth Science Data for all: EOS and the EOS Data and Information System. *Photogrammetric Engineering and Remote Sensing*, 60:277–285, 1994.

[VR94] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-Plus.* Springer-Verlag, 1994.

[WM98] C. J. Willmott and K. Matsuura. Global Air Temperature and Precipitation: Regridded Monthly and Annual Climatologies (version 2.01). *Center for Climatic Research, Dept. of Geography, Univ. of Delaware.*, 1998.

[YWK+99] R. Yang, C. Wang, M. Kafatos, X. Wang, and T. El-Ghazawi. Remote Data Access via SIESIP Distributed Information System. In Z. Meral Ozsoyoglu, Gultekin Ozsoyoglu, and Wen-Chi Hou, editors, *Proceedings of the 11th International Conference on Scientific and Statistical Database Management*, page 284. IEEE, Computer Society, 1999.

[YYKW00] R. Yang, K. Yang, M. Kafatos, and X. S. Wang. Value Range Queries on Earth Science Data via Histogram Clustering. In Kathleen Hornsby and John F. Roddick, editors, *Proceedings of International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining (TSDM2000)*, 2000.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD Rec.*, 25(2):103–114, 1996.

Chapter 12

SPATIAL DATA MINING FOR CLASSIFICATION, VISUALISATION AND INTERPRETATION WITH ARTMAP NEURAL NETWORK

Weiguo Liu, Sucharita Gopal, and Curtis Woodcock

Abstract Accurate global land cover information is required for many aspects of global change research. Remote sensing provides the only viable basis for the production of this information. This paper reports research undertaken as part of the MODIS effort to map the land cover of North America using the ARTMAP neural network. The main objective is to design a system called ART-VIP (ART for Visualisation and Image Processing) that integrates the ARTMAP neural network algorithm into a standard public domain image processing software, and to help users analyse and interpret the dynamics of the ARTMAP neural network with scientific visualisation tools. The provision of public domain software and methodologies facilitates the use of the ARTMAP neural network architecture for other land cover classification problems.

Keywords: ART-VIP, ARTMAP, Neural Network, Visualisation, Interpretation, and MODIS

1. Background

1.1 Global Land Cover Mapping

Global land cover is essential to many kinds of global change research and environmental applications. Land cover consisting of inherent embedded information on vegetation, is perhaps the most fundamental characteristic of a place. It is an integrated expression of the physical, climatic and biotic environment as well as the history of land use by humans. As such, people have been interested in mapping land cover and vegetation for centuries. In

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 201–221.
© 2001 *Kluwer Academic Publishers.*

this context, land cover directly influences whole suites of processes such as the exchange of mass and energy between the surface and atmosphere. Similarly, land cover is necessary for deriving many essential surface parameters.

In the past few years, land cover mapping at continental and global scales has been revolutionized by the use of remote sensing imagery from the AVHRR satellites. NASA's new satellite called MODIS (Moderate Resolution Imaging Spectroradiometer) (Strahler et al., 1999) will view the entire Earth's surface every 1 to 2 days, acquiring data in 36 spectral bands. The spatial scale and data volumes of this classification problem dictate the need for classification algorithms that minimize requirements for human intervention and automated classification. Currently, Boston University is developing global land cover mapping algorithms for use with data from MODIS.

The goal of the MODIS land cover group at Boston University is to generate automated maps of global land cover using supervised classification techniques including decision trees and neural networks (ARTMAP). The algorithm development and validation efforts are based on a network of test sites chosen to represent major global biomes and cover types. Pre-launch efforts have focused in North America for which temporal sequences of TM and AVHRR imagery data are available. The main objective of this paper is to design a system called ART-VIP (ART for Visualisation and Image Processing) that integrates the ARTMAP neural network algorithm into a standard public domain image processing software (called Image Processing Workbench or IPW), and to help users analyse and interpret the dynamics of the ARTMAP neural network with scientific visualisation tools. The provision of public domain software and methodologies facilitates the use of the ARTMAP neural network architecture for other land cover classification problems.

1.2 Neural Networks in Remote Sensing

Neural networks have proven to be the most significant improvement in information extraction in remote sensing in the last 10 years. The classification of remotely sensed data using artificial neural networks began appearing in the remote sensing literature about ten years ago. Since then, examples and applications have become increasingly common. Remotely-sensed data sets processed by neural network-based classifiers have included images acquired by the Landsat Multispectral Scanner (MSS) (Benediktsson et al., 1990; Lee et al., 1990) Landsat TM (Yoshida et al., 1994; Gopal et al., 1996) synthetic aperture radar (Hara et al., 1994) SPOT HRV (Tzeng et al., 1994) AVHRR (Gopal et al., 1994) and aircraft scanner data (Benediktsson et al., 1993). A number of these studies have also included ancillary data e.g., topography (Carpenter et al., 1997) and texture (Bischoff et al., 1992). Many studies have been directed toward recognition of land cover classes, which have ranged from broad life-form categories (Hepner et al., 1990) to floristic classes. Most use a supervised approach, but unsupervised classification using self-organizing neural networks has also been attempted (Hara et al., 1994).

As the number of applications of neural network classification increases, it is increasingly clear that neural network-based classification can produce more accurate results than conventional approaches in many remote sensing classification problems. The reasons include: (a) neural network classifiers, which make no a *priori* assumption about data distributions, are able to learn nonlinear and discontinuous patterns in the distribution of classes; (b) neural networks can readily accommodate collateral data such as textural information, slope, aspect and elevation; and (c) neural networks are quite flexible and can be adapted to improve performance for particular problems. There are some limitations in using neural network approach in terms of computation time (Paolo and Schowengerdt, 1995) that has led to integration of neural network and statistical approaches (Wilkinson et al., 1994).

The bulk of neural network classification work in remote sensing has used multilayer feed-forward networks that are trained using the backpropagation algorithm based on a recursive learning procedure with a gradient descent search. However, this training procedure is sensitive to the choice of initial network parameters and to overfitting (Fischer et al., 1997). The use of neural networks utilizing Adaptive Resonance Theory (ART) (Carpenter and Grossberg, 1991), can overcome these problems. Networks organized on the ART principle are stable when learning proceeds, while at the same time they are plastic enough to learn new patterns. Fuzzy ARTMAP is a supervised learning system that synthesizes fuzzy logic and ART models. Some recent studies highlight the utility of the fuzzy ARTMAP architecture and its application to land cover classification, (Carpenter et al., 1997 and 1998b) mixtures vegetation mapping (Carpenter et al., 1998a), and change detection (Pax-Lenney et al., 1999).

2. Adaptive Resonance Theory and ARTMAP Neural Networks

The ART family of models was developed by Carpenter and Grossberg (Carpenter and Grossberg, 1991; Carpenter et al. 1991a; 1991b; 1992). ART is a match-based learning system. The major feature of ART is its ability to solve a stability-plasticity dilemma. Too often, learning a new pattern erases or modifies previous learning. This may not be a problem for a fixed set of training vectors, when the network can eventually learn all the patterns if it is allowed to cycle through the training data repeatedly. Such conditions are rarely met in real world contexts, however, since the network is more often exposed to a changing environment and rarely exposed to same training vector twice. For example, in a multi-layer perceptron network such as backpropagation, a new training vector added to a previously trained network may have catastrophic side effects, severely disrupting learned weights. This is a serious limitation, since retraining can be computationally expensive or impossible. In contrast to the conventional models, ART networks and models maintain the stability of categorizing previously learned patterns, while remaining plastic enough to learn new pattern categories. ART models form stable recognition categories of optimal size when given analog or binary inputs in a random order.

2.1 ART Neural Networks

The description of ART networks given below (Carpenter and Grossberg, 1991; Carpenter et al. 1991a; 1991b; 1992) outlines the essential features of adaptive resonance theory. A minimal ART1 module consists of a two-level network of interconnecting neurons: a comparison level (F_1) and a recognition level (F_2) (see Figure 1). An input signal in the form of a binary vector (F_0) presented to F_1 is propagated forward to F_2. F_1 and F_2 are connected with both feedforward connections (F_1 to F_2) and feedback (F_2 to F_1) connections. Long-term memory in encoded in both these feedforward and feedback connections. F_2 nodes interact with each other by lateral inhibition. The result is a competitive winner-take-all response producing an F_2 activity pattern in which only the node associated with a single category is significantly activated. The corresponding learned weight vector is then propagated backward to the F_1 level where it is compared with the original input vector. If the two patterns are close according to a matching criterion determined by an ART 'vigilance' parameter, 'resonance' occurs and long-term memory is altered to incorporate the new observation. If the two patterns differ significantly, the ART module enters a search mode. During this phase, the network attempts to find a new category node in the F_2 layer for the current input vector **A**. The present active node in F_2 is disabled and a second category node is selected. The new weight vector is then propagated back to the F_1 level and compared with the input vector as before. If the two match, then resonance proceeds. If not, the second F_2 node is disabled and another attempt is made to find a good match. The process repeats until eventually a matching F_2 node is found or a new category is established.

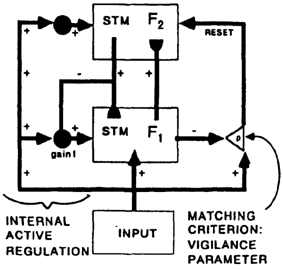

STM F2
RESET
STM F1
gain I
INPUT
INTERNAL ACTIVE REGULATION
MATCHING CRITERION: VIGILANCE PARAMETER
ρ
+
-

Figure 1. ART1 Module (Carpenter and Grossberg, 1991)

The binary ART1 module described above is not an associative memory system (Carpenter and Grossberg, 1991). However, a minimal ART1 architecture embedded in a larger system can perform as an associative memory system. Fuzzy ARTMAP (Carpenter et al., 1992) incorporates fuzzy logic (Zadeh, 1966) in its ART1 modules and has fuzzy set theoretic operations instead of ART1's binary set theoretic operations.

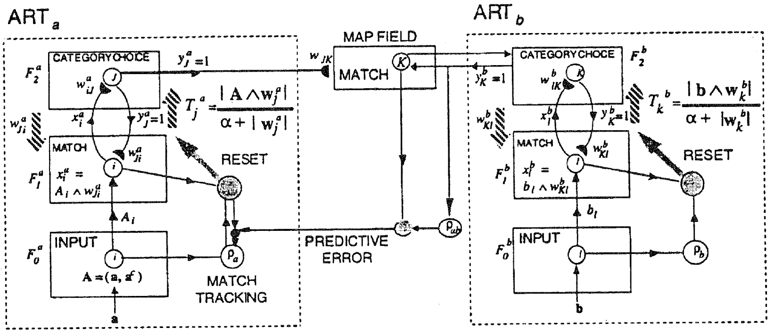

ART a
ART b
MAP FIELD
MATCH
CATEGORY CHOICE
MATCH
INPUT
RESET
MATCH TRACKING
PREDICTIVE ERROR
F_2^a
F_1^a
F_0^a
F_2^b
F_1^b
F_0^b
$y_J^a = 1$
w_{JK}
$y_K^b = 1$
$T_j^a = \frac{|A \wedge w_j^a|}{\alpha + |w_j^a|}$
$T_k^b = \frac{|b \wedge w_k^b|}{\alpha + |w_k^b|}$
$x_i^a = A_i \wedge w_{Ji}^a$
$x_I^b = b_I \wedge w_{KI}^b$
$A = (a, a^c)$
ρ_a
ρ_b
ρ_{ab}
A_i
b_I
a
b

Figure 2. The architecture of fuzzy ARTMAP (Carpenter et al., 1992)

The basic architecture of fuzzy ARTMAP is shown in Figure 2. It consists of a pair of fuzzy ART modules, ART_a and ART_b, connected by an associative learning network called a Map Field. The other component of this architecture is a controller that uses a minimum learning rule to conjointly minimise predictive error and maximise code compression or predictive generalisation (Carpenter and Grossberg, 1991; Carpenter et al. 199a; 1991b). It enables the system to operate in real time and determine the number of hidden units (or recognition categories) needed to meet the accuracy or matching standards. The 'hidden units' in ART_a represent learned recognition categories. In the training phase, ART_a and ART_b modules of the system are presented with a stream of input a^p and desired output pairs b^p respectively. The two modules classify the a^p and b^p vectors into categories and the map field makes the association between ART$_a$ and ART$_b$ categories. A mismatch between the actual b^p and predicted b^p causes a memory search in ART$_a$. A mechanism called match tracking then raises the ART$_a$ vigilance ρ_a by the minimum amount necessary to trigger a memory search. This can lead to a selection of a new ART$_a$ category that is a better predictor of b^p. Between learning trials, the vigilance relaxes back to its baseline value ρ_a. Match tracking therefore sacrifices only the minimum amount of generalisation needed to correct a predictive error. Fast learning and match tracking enable fuzzy ARTMAP to learn to predict novel events while maximising code compression and preserving code stability.

2.2 Fuzzy ARTMAP Algorithm

The fuzzy ARTMAP algorithm uses a choice parameter, $\alpha > 0$; a learning parameter, $\beta \in [0,1]$; and a vigilance parameter, $\rho \in [0,1]$. We summarize the fuzzy ART algorithm by describing how procedures including category choice, reset and learning are executed. The *choice function* in fuzzy ARTMAP denoted by T_j is defined for each input **A** and F_2 node j as follows:

$$T_j(A) = \frac{|A \Lambda W_j|}{\alpha + |W_j|}, \tag{12.1}$$

Where the fuzzy AND operator ' Λ ' is defined by

$$(P \Lambda Q)_i = \min(P_i, Q_i), \tag{12.2}$$

and the norm $|.|$ is defined by

$$|P| = \sum_{i=1}^{M} p_i, \tag{12.3}$$

for any M-dimensional vectors p and q.

When any given time, only one F_2 node can be active the system is said to have made a category choice. Let the category choice be indexed by **J** and let $T_j(A)$ in the above equation be written as T_j for notational simplicity when the input **A** is fixed. Then

$$T_J = \max\{T_j, j=1,\ldots,M\}. \tag{12.4}$$

Note that if more than one T_j is maximal, the category **J** with the smallest index is chosen. The nodes become committed in order **j** = 1, 2, 3, $\cdots$. The equation governing the F_1 activity vector x is

$$x = \begin{cases} A & \text{if } F_2 \text{ is not active,} \\ A \wedge W_J & \text{if } J_{th}\, F_2 \text{ is active.} \end{cases} \tag{12.5}$$

Also for the Jth category to be chosen, $y_J = 1$ and $y_j = 0$ for $j \neq J$.

Resonance or Reset Operation: Resonance occurs if the match function $\frac{|I \wedge W_J|}{|I|}$ of the chosen Jth category exceeds the vigilance criterion defined by ρ. A mismatch reset occurs when $\frac{|I \wedge W_J|}{|I|}$ is less than ρ. So long as A remains constant, the same category J that was already selected as a category during search cannot then be selected.

Learning: Once search terminates, the weight vector W_J is updated according to the following equation

$$W_J^{new} = \beta (A \wedge W_J^{old}) + (1-\beta) W_J^{old}. \tag{12.6}$$

In this equation fast learning corresponds to setting β to 1. A mathematical and geometrical interpretation of fuzzy ART as well details of system dynamics and algorithm are presented in (Carpenter and Grossberg, 1991; Carpenter et al. 1991a; 1991b; 1992).

2.3 ARTMAP Categories

During training, the weights of the F_2 layer of nodes get created and updated as a result of learning. Initially, at the start of training there is no node in the F_2 layer. The first input vector will trigger the generation of the first category (F_2 node), which represents the properties of input vector. Based on equations 6, each input vector is matched with the existing F_2 categories. A new category is created only if no existing category can match the statistics of the input vector based on learning and other parameters. When there is a match, the category satisfying the match function will use the current input vector to refine its weight value in order to incorporate more general (spectral) characteristics for this category. The weight vector refining process can be regarded as broadening the categorization. Matching process will generate more categories and will ensure that the similarity in spectral space is transferred and represented in the category's weight vector.

Fuzzy ARTMAP may use several categories to represent one class in order to capture the spectral variance in inputs relating to the class. Fuzzy ARTMAP categories represent the inter-class and intra-class variability among classes. To summarize, (1) each F_2 category (or node) in fuzzy ARTMAP extracts and generates common spectral properties from a cluster of input vectors. (2) Fuzzy ARTMAP can represent the intra-class variability since the variance in spectral space of any class will trigger the generation of many categories in weight space. Thus, one class will be represented by several F_2 categories. (3). The map field associates the categories with each class. The many-to-one connection facilitates classification. (4). Different F_2 categories of same class have certain similarity. Usually, they share the same region in neural network (hyper-dimensional) weight space.

3. Design and Implementation of ART-VIP

The main limitation of any neural network classifier in remote sensing is the availability of the software that can be linked to a remote sensing software package such that data analysis and classification can be done efficiently. Recently commercial packages such as PCI DEASI/PACE Imaging Pro Kit (EPIP) have included backpropagation neural networks. Users may not have access to expensive software or have knowledge on how to set up various crucial parameters. In addition, there is a gap between research and software implementation. At Boston University, we have used Image Processing Workbench (IPW), initially developed at the University of California, Santa Barbara, by James Frew (Frew, 1990) for analysis and processing of remote sensing data. Much of the source code of IPW is in the public domain; a version of fuzzy ARTMAP neural network software is also available as public domain software. We have modified the existing ARTMAP software considerably and integrated it into IPW platform to develop ART-VIP.

3.1 The Goals of Developing ART-VIP

Simplicity: A daunting task in the use of most neural network classifications is the parameters. Users have to be aware of the implications of these parameter settings since they can affect results quite dramatically. For example, in backpropagation networks, there is no rule on how best to select the number of hidden units, except trial and error. The use of ARTMAP overcomes this problem since the number of hidden units is developed by the algorithm. The most critical parameter is ARTMAP is vigilance. All the other parameter values could be set up such that the user need not select any of their values. Thus in the integrated system, the user can use default values and still get reasonable classification results.

Extensibility and portability: ART-VIP is usable in any standard UNIX and C environments. The system operates on generic multi-band images.

However, there is nothing about the software that restricts its use to particular data source (sensor, data system, etc.) since its internal data format is extremely simple. Therefore, ART-VIP is designed for growth over time, and eventual use with MODIS and other sensor data.

Flexibility: ART-VIP is a flexible system such that users can adjust the parameters based on their needs including the types of data or the types of problems being examined. New developments in ARTMAP algorithm can also be easily added.

Data import and export: A fundamental design decision in ART-VIP has been to facilitate the importation and exportation of data between ARTMAP and IPW. ARTMAP has to read IPW image data and classify it, and send the classified data back into IPW to display the whole image. Thus ART-VIP allows for data to be communicated easily between ARTMAP and IPW software.

Visualisation: Most neural networks are treated as "black boxes" -- meaning that the nature of the signal used from the input variables and its relationship to the output is usually unknown. From a remote sensing perspective, neural networks will prove most useful if their internal behavior can be understood. Also, the kinds of uses for neural networks expand if their internal workings become less opaque. Hence, we have used powerful visualisation tools in ART-VIP that helps the users to analyze some critical dynamics of ARTMAP algorithm.

3.2 The Modules of the Integrated System

There are four modules in ART-VIP, ARTPREP, ARTTRAIN, and ARTCLASS and VISUALIZE. Considering both the dynamics of fuzzy ARTMAP and software development strategies, we represent the first three modules in Figure 3.

ARTPREP: This module deals with data preparation. ARTPREP can read the IPW format image and extract the spectral information of a training data set and its associated label information for supervised training. Since fuzzy ARTMAP requires input data to be normalized between the values of 0 and 1, there is a built-in normalization function in ARTPREP. The default setting for normalization utilizes a simple linear normalization and the program can adaptively set the initial range based on the input data. However, there are two options by which users can modify the normalization process to satisfy their specific requirements. One way to alter the normalization is to change the header information on the IPW image header that includes linear quantization information. A second method is to create a specific normalization file tailored for a specific dataset. ARTPREP can normalize the input data according to the linear quantization header or the normalization file provided

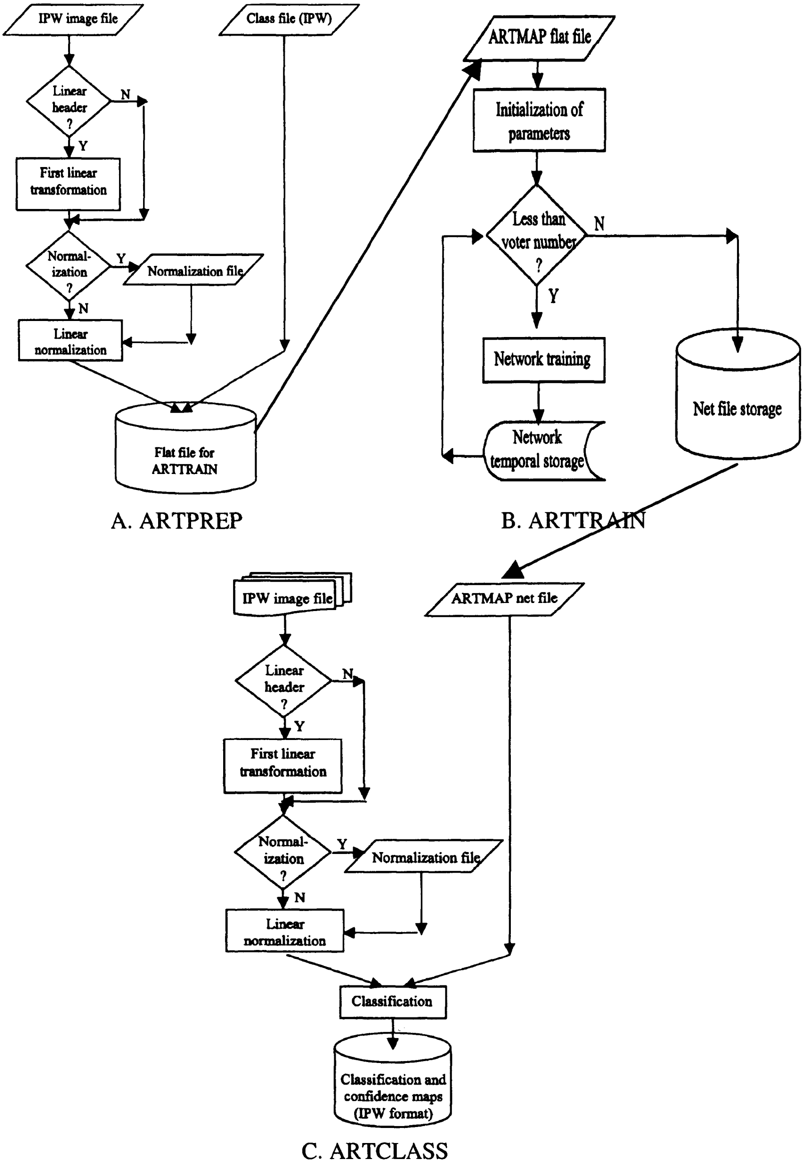

IPW image file
Linear header ?
N
Y
First linear transformation
Normal-ization ?
Y
Normalization file
N
Linear normalization
Class file (IPW)
Flat file for ARTTRAIN
A. ARTPREP
ARTMAP flat file
Initialization of parameters
Less than voter number ?
N
Y
Network training
Network temporal storage
Net file storage
B. ARTTRAIN
IPW image file
Linear header ?
N
Y
First linear transformation
Normal-ization ?
Y
Normalization file
N
Linear normalization
ARTMAP net file
Classification
Classification and confidence maps (IPW format)
C. ARTCLASS

Figure 3. A flow-char showing modules of integrated ARTMAP system

by users, or a combination of both. A combination use of both is useful when images of different regions or dates are to be classified together.

ARTTRAIN: There are many parameters of fuzzy ARTMAP, including vigilance, choice, and learning rate. Most of the parameters can be set to default values. The default value of vigilance can be set to 0 to obtain the maximum generalization (or coarse categories). The choice parameter is set to a small positive value. The default value of learning rate is 1 for fast learning. Another important parameter is maximum number of training epochs. The software can automatically terminate the learning if the learning accuracy reaches 100 percent training accuracy. The voting strategy in fuzzy ARTMAP is important. Hence the user has the option of choosing the number of voters. Each voter in the scheme has an associated, trained neural network. The trained network can be used over again. This concept is very different from conventional image classification in remote sensing. This feature is one of the benefits of the modular structure.

ARTCLASS: This module uses the same technique as ARTPREP to read and normalize the image to be classified. One useful option of ARTCLASS is the confidence file, which can show the frequency of voters of each class for a pixel. The final confidence map is a standard IPW format image that can display the confidence in terms of votes for a class as well as a single class at a pixel. This image can be used for the uncertainty analysis.

3.3 Visualisation in ART-VIP

We have written several MATLAB programs for visualisation and numerical analysis of basic ARTMAP dynamics. These programs can display 2-D or 3-D images of ARTMAP dynamics during training and testing. They can help the user to visualize or analyze the training process and classification results obtained from the ARTMAP network.

F_2 Evolution Plot: The evolution and updating of weight vector of a F_2 category can be useful to show the dynamics and properties of fuzzy ARTMAP algorithm. This is shown as a sequence of plots so that the reader can visualize the initial and final plots of a F_2 category as well as its evolution.

ARTMAP Box or Hyper-Box Plots: Visualisation may help in understanding how the input features are represented internally by fuzzy ARTMAP. This type of plot shows network weight space corresponding to two or more spectral inputs in 2 or 3 dimensions for each output class. In this paper we have used 2 dimensional plots of two or more spectral inputs. The geometry of the plot is defined by stored weights of input vector. Carpenter et al. (1997) has used two-dimensional plots where the size and number of boxes provide insights into ARTMAP learning process. In this paper, a new type of ARTMAP visualisation plot is introduced. The first and second largest F_2 category boxes associated with each output are drawn to understand the differences between F_2 categories.

ARTMAP Testing Plots: This program can generate testing plots in terms of circle graphs of individual F_2 nodes that show when errors in classification are made.

4. ART-VIP Application in Global Landcover Mapping with MODIS Dataset

ART-VIP was developed in the context of MODIS global landcover mapping at Boston University. Its development was dictated by user needs to understand the classification process including nature of training data and spatial distribution of errors. The modular structure of the ART-VIP system allows users to understand the classification process in a systematic way. The dataset used in the present study consists of a time series of 36 ten-day AVHRR (Advanced Very High Resolution Radiometer) maximum value normalized difference vegetation index (NDVI) composites provided by EDC for the period Feb. 95 through Jan. 96. The final training data and associated class labels were compiled by the MODIS land cover team at Boston University. A total of over 1000 training sites were used in this study. Land cover maps were made of North America using decision trees (Friedl et al., 1999) and fuzzy ARTMAP neural network. Both decision trees and fuzzy ARTMAP classifiers were first trained on 80% of data, and 20% of data was used to test the accuracy of classification.

The ARTMAP is trained using the composite NDVI data for 12 months. The order of presentation of training set affects the training of the system and hence 5 different splits were made such that each pixel is tested once independently. The output classes are defined by the International Geosphere and Biosphere program (IGBP) and consist of 16 classes. Out of the 16 classes, 11 classes are of natural vegetation, 2 classes are of mosaic lands, and 3 classes are of non-vegetated lands.

ART-VIP aids in the visualization of ARTMAP's training process that is useful for interpreting the quality of training dataset and classification results. ART-VIP's design features and its integration with a widely used remote sensing software (IPW) are useful in the landcover mapping process. In next section, we will discuss the function of visualization with ART-VIP in MODIS landcover mapping project.

5. Visualisation Plots Related to MODIS Data Set

In this section, the basic dynamics of ARTMAP will be interpreted using visualisation tools in ART-VIP.

F_2 Evolution Plot: In Figure 4, each step, or snapshot, describes the weight updating process by which a single F_2 category can be transformed. The red line represents the weight's initial value before input vector presentation; the blue line represents the current input vector; and the green line shows the final status of the weight after the incorporation of the input vector. As described in Section 2.1, the initial value of weight in a new F_2 category is 1 and the weight update is based on Equation 6. The values of the weight vector always

decrease or remain stable during the learning process. Step one in Figure 4 shows that the initial value of weight vector is (1 ,..., 1) is modified in the next step to match the input vector. In the second step, the weight vector is exactly the same as the input vector. For the third step, the weight value in several dimensions is larger than the value of the input vector. So the weight value of the F_2 node remains stable in these dimensions but decreases to the value of the input vector in other dimensions. The weight vector captures the properties of two input patterns enabling recognition of these two patterns as well as the patterns very similar to them. In Step 5, the weight vector does not change suggesting the input feature properties have been included in the current weight vector. Steps 6, 7, and 8 show that the weight vector has not changed with an addition of new inputs. The final step (step 9) shows the weight vector associated with a F_2 category that represents the common characteristics of a cluster of input vectors. Users can display any F_2 category to analyze the updating process and final representation in order to understand the transformation of spectral data from spectral space to weight space.

ARTMAP Box or Hyper-Box Plots: ARTMAP uses complement coding to normalize the input vector automatically. Thus, for the MODIS dataset, the input vector, (including complement coded information), consists of 24 dimensions. Hence, the associated weight vector of each category has24 dimensions, consisting of two parts, U_j representing the original weight vector, and V_j representing the complement code weight vectors. Weight vectors can be represented geometrically as boxes in the M-dimensional space of the input vector using both the original and complement weight vector. Carpenter et al. (1997) in several studies described the geometry of boxes using 2 dimensions. In this study, there are twelve input vectors and 17 output classes. We provide 3 types of visualisation tools to analyze the categories. Figure 5 shows ARTMAP box plot where the categories are represented as rectangular boxes using the August NDVI (8th input vector) and January NDVI (1st input vector) in the x-axis and y-axis for 15 classes to illustrate the geometry of fuzzy ARTMAP learning. For any of category J, the left-bottom coordinate of the rectangular box is (U_{jx}, U_{jy}), the top-right coordinate is ($1-V_{jx}$, $1-V_{jy}$). These plots show that certain classes such as 1 and 12 have more internal categories compared with class 15 and 3. In addition, the size of boxes are bigger for classes 1 and 12 compared with class 3 and class 15 suggesting that the former may be more heterogeneous compared with the latter.

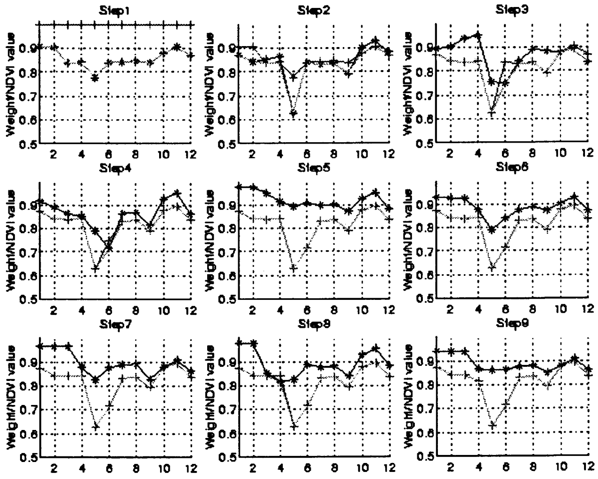


Figure 4. A plot displaying evolution of a F2 node

Hyper-Box Plots: The box plots described above cannot represent the information of twelve dimensions. For the sake of visual interpretation, Figure 6 shows the input feature space represented as a two-dimensional plot, where the x-axis represents the 12-input feature vector and the y-axis represents weight value of the corresponding categories (F_2 nodes). Figure 6 shows the status of all original weight vectors for each class after training. Each line represents the weight vector of one category. More than six hundred categories were generated during the training process. Hence it is difficult to visualise them together.

Hyper-box Plots for the two largest Categories: For sake of visualisation, for each class, the two biggest categories (based on the number of training input vectors contributing to the generation of the category) are represented in Figure 7. The top half of the figure is the hyper-box of the largest category and the lower half is the hyper-box of the second largest category for each class. The red lines represent the U_j (original weight vector) and pink lines represent the vector whose value equals to 1-V_j (complement weight vector), the blue lines represent the input feature vector generating the category J. The size and location of the hyper-box illustrates the properties of the category J in weight space. Each hyper-box stores all the attributes common to all the input vectors inside of it. The hyper-box represents the intra-class similarity inside a category. For each class, the number and distribution of categories can be seen to illustrate the variance inside the class.

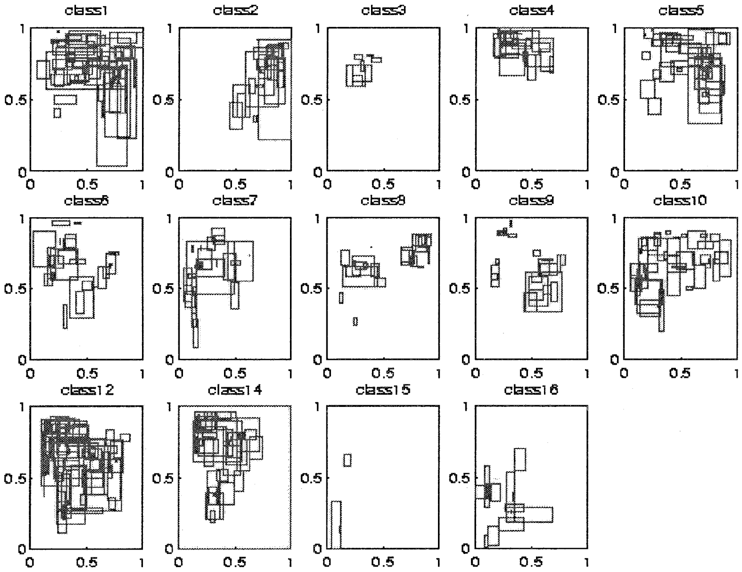


Figure 5. ARTMAP box plots of MODIS classes

Thus Figures 5, 6 and 7 together provide some useful information. In Figure 5, the number of rectangular boxes for class 1 and 12 are much larger than that of the other classes. The size of boxes for these 2 classes are not regular, dispersed in location, and the total area of the boxes is roughly more than one-half of the weight space. This illustrates that the variance inside the two classes is high and they can be grouped into several sub-classes. On the other hand, classes 3 and 15 show a different picture. The number of their boxes is much less and the distribution in weight space is more aggregated. This shows the two classes are homogeneous. This is reinforced in Figure 6 which shows the distribution of categories related to each class. Figure 7 highlights the previous results that classes 1, 2, and 12 have several sub clusters. Each cluster may have generated one or more categories in weight space.

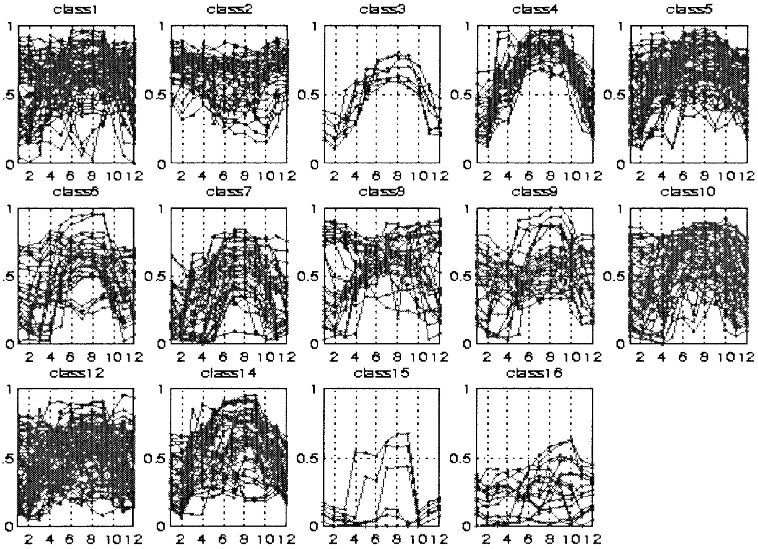


Figure 6. ARTMAP hyper-box of MODIS classes

The size of a hyper-box illustrates the generality of the category. A larger box is more suggestive of a general category. This kind of category captures more variance in the input data. Some category boxes are so small that they tend to represent a single input vector. For example, the second biggest category of class 15 in Figure 7 illustrates that its input vector is so specific that no other input is similar to it. Figure 7 also shows that different categories have certain overlap regions which means that one input vector can be placed into more than one category, i.e. the largest categories of class 1 and 2 overlap considerably.

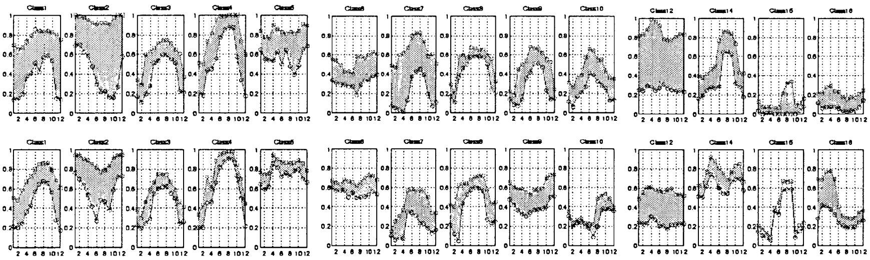


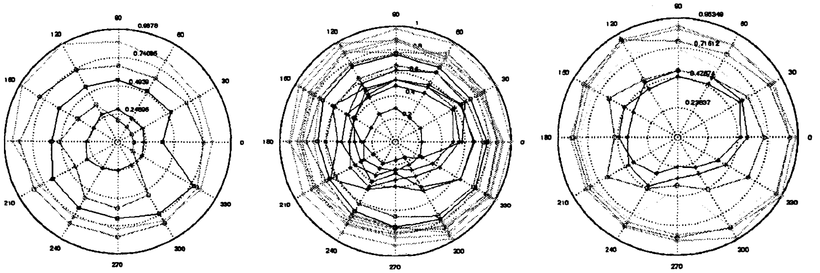
Figure 7. ARTMAP hyper-box for two largest categories in weight space

ARTMAP Testing Plots: The ARTMAP weight vector is called a *template* that stores the attributes of the category (Huang et al., 1995). Let I be an input vector presented to fuzzy ARTMAP and the arbitrary template denoted by W_j. Let the index of the input vector I be i representing the spectral feature and the corresponding component of template W_j be W_{ji}. Two types of templates can occur in fuzzy ARTMAP with complement coding: ***subset templates*** and ***mixed templates*** (Huang et al. 1995). A template W_j is a ***subset template*** for an input vector if any component of W_j is smaller than or equal to its corresponding component in I. A *Mixed template* is produced when some of the W_j components are larger than their corresponding components in the remaining and I is smaller or equal to its corresponding components in I. In the fuzzy ARTMAP architecture, when an input vector I is presented at the F_1 layer, a category (node in F_2 layer) is chosen according to the following rules:

Rule 1: Among all *subset templates*, the category with largest template will be selected first.

Rule 2: If $|I \wedge W_j| > |W_j|$, a *mixed template* W_j is accessed prior to the *subset template* W_J.

Note that template here refers to the original weight vector. Each template (category) can be visualized as the bottom line of a hyper-box in neural network weight space. Thus if the input vector fall in several hyper-boxes, the smallest hyper-box that has the minimum distance to the input vector will be chosen first. This can be visualized using a circle diagram. The axes of circles represent the weight value of a category (F_2 node) along each input vector. In Figure 8a, the black line represents the input vector. The input vector was classified correctly into one hyper-box whose upper and lower bounds are shown in red and pink respectively. Note that the input vector could have also fell into other hyper-boxes shown by blue and green lines. Based on the rule 1, the input vector is classified into the small hyper-box correctly. Figure 8b shows a more complicated picture since the input vector shown in black falls into more than 2 hyper-boxes but is classified correctly. Figure 8c shows the input vector drawn in yellow falls into 3 hyper-boxes but there is a misclassification. The hyper-box selected is not the correct (using Rule 1). Figure 8c illustrates the conditions in which typical ARTMAP misclassification occurs.



a b c

Figure 8. ARTMAP testing plots illustrating correct and incorrect classification

Rule 2 is interpreted using Figure 9. Consider a 5-dimensional simulated input vector denoted by I and one subset template W_1 and one mixed template W_2. According to equations in Section 2.1, the winning category should be the larger one of $\frac{|I \wedge W_2|}{|\alpha + W_2|}$ and $\frac{|I \wedge W_1|}{|\alpha + W_1|}$. W_1 is the subset template of input vector I, $|I \wedge W_1| = |W_1|$. $|I \wedge W_2| > |W_1|$, so $|W_2| > |W_1|$. This results in $\frac{|I \wedge W_2|}{|\alpha + W_2|} > \frac{|I \wedge W_1|}{|\alpha + W_1|}$. Thus, a mixed template is chosen as the category of input vector I. That may lead to misclassification.

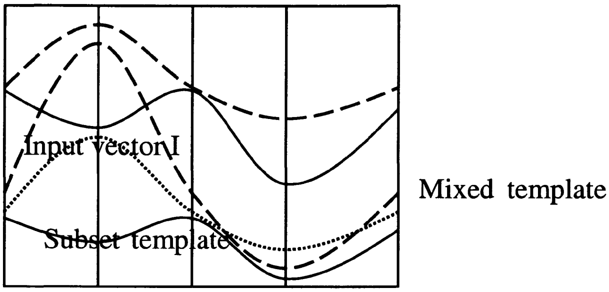


Figure 9. A five dimension simulated data illustrating rule 2

6. Conclusion

This paper presents the ART-VIP system that integrates the ARTMAP neural network model with image processing software. The advantage of ART-VIP is that it leads to more efficient and automated image classification using remote sensing data. It's visualisation module helps users to interpret the dynamics of ARTMAP neural network model, thus 'demystify' the black box of neural network. Compared with existing remote sensing commercial software packages, ART-VIP visualization tools help users to visualize decision surfaces both in feature as well as spectral space. In addition, the modular design of ART-VIP allows for step-by-step analysis of the neural network classification process. Preliminary tests of the ART-VIP system in a classroom setting shows that the seamless integration of the neural network algorithm in an existing image processing software and visualization of the neural network classification process aids in users understanding. The ART-VIP system is presented in the context of the MODIS land cover project. But its utility extends to other remote sensing problems. ART-VIP is available as public domain software developed in C language in Unix environment.

Acknowledgements

This research is supported by the following grants - NSF SES-9973474, NSF SBR-9513889 and NASA NAS5-31369. The preliminary version of this paper was published in the 9th international Symposium on Spatial Data Handling, 10-12 August, 2000, Beijing, China.

References

[BS90] J. Benediktsson, P. Swain, O. Ersoy, Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data, *IEEE Transactions on Geoscience and Remote Sensing*, 28, 540-552, 1990.

[BS93] J. Benediktsson, P. Swain, O. Ersoy, Conjugate-gradient Neural Networks in Classification of Multisource and Very-high-dimensional Remote Sensing Data, *International Journal of Remote Sensing*, 14, 2883-2903, 1993.

[BS92] H. Bischoff, W. Schneider, A. Pinz, Multispectral Classification of Landsat-images Using Neural Networks, *IEEE Transactions on Geoscience and Remote Sensing*, 30, 482-490, 1992.

[CG91a] G. Carpenter and S. Grossberg, *Pattern Recognition by Self-organizing Neural Networks*. Cambridge, MA: MIT Press, 1991.

[CG91b] G. Carpenter, S. Grossberg, J. Reynolds, ARTMAP: Supervised Real-time Learning and Classification of Nonstationary Data by A Self-organizing Neural Network, *Neural Networks*, 4, 565-588, 1991.

[CG91c] G. Carpenter, S. Grossberg, D. Rosen, Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by An Adaptive Resonance System, *Neural Networks*, 4, 759-771, 1991.

[CG92] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, D. Rosen, Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps, *IEEE Transactions on Neural Networks*, 3, 698-713, 1992.

[CG97] G. Carpenter, M. Gjaja, S. Gopal, C. Woodcock, ART Networks in Remote Sensing, *IEEE Transactions on Geoscience and Remote Sensing*, 35(2), 308-325, 1997.

[CG98] G. Carpenter, S. Gopal, S. Macomber, S. Martens, C. Woodcock, A Neural Network Method for Mixture Estimation for Vegetation Mapping, *Remote Sensing of the Environment*, 70, 138-152, 1998.

[CF98] G. Carpenter, J. Franklin, S. Gopal, S. Martens, C. Woodcock, Mapping Vegetation Lifeforms and Species Associations Using Landsat, Terrain and Geographic Location Based on Fuzzy ARTMAP, *Remote Sensing of the Environment*, 70, 1998.

[FG97] M. Fischer, S. Gopal, P. Staufer, K. Steinocher, Evaluation of Neural Pattern Classifiers for A Remote Sensing Application, *Geographical Systems*, 1997.

[FJ90] J. Frew, *The Image Processing Workbench*, PhD Thesis, Department of Geography, University of California, Santa Barbara, CA, 382 pp.

[FB99] M. Friedl, C. Brodley, A. Strahler, Maximizing Land Cover Classification Accuracy Produced by Decision Trees at Continental to Global Scales, *IEEE Transactions on Geoscience and Remote Sensing*, 37, 969-977, 1999.

[GS94] S. Gopal, D. Sklarew, E. Lambin, Fuzzy Neural Networks in Multi-temporal Classification of Landcover Change in The Sahel, in *Proceedings of the DOSES Workshop on New Tools for Spatial Analysis*, Lisbon, Portugal, DOSES, EUROSTAT, pp. 55-68. ECSC-EC-EAEC: Brussels, Luxembourg, 1994.

[GW96] S. Gopal, and C. Woodcock, Remote Sensing of Forest Change Using Artificial Neural Networks, *IEEE Transactions on Geoscience and Remote Sensing*, 34, 398-404, 1996.

[HA94] Y. Hara, R. Atkins, S. Yueh, R. Shin, J. Kong, Application of Neural Networks to Radar Image Classification, *IEEE Transactions on Geoscience and Remote Sensing*, 32, 100-111, 1994.

[HL90] G. Hepner, T. Logan, N. Ritter, N. Bryant, Artificial Neural Network Classification Using A Minimal Training Set Comparison of Conventional Supervised Classification, *Photogrammetric Engineering and Remote Sensing*, 56, 469-473, 1990.

[LW90] J. Lee, W. Weger, S. Sengupta, R. Welch, A Neural Network Approach to Cloud Classification, *IEEE Transactions on Geoscience and Remote Sensing*, 28, 846-855, 1990.

[PS95] J. Paola, and R. Schowengerdt, A Detailed Comparison of Backpropagation Neural Networks and Maximum Likelihood Classifiers for Urban Landuse Classification. *IEEE transactions on Geoscience and Remote Sensing*, Vol. 33, No. 4, pp. 981-996, 1995.

[PW00] M. Pax-lenney, C. Woodcock, S. Macomber, and C. Song, Identification of Conifer Forests in Landsat TM: A New Look at Classification Generalization, Manuscript in Review, *Remote Sensing of the Environment*, 2000.

[SA99] Strahler, A.H. et al. *Modis Landcover and Land cover change product*, Version 5, ATBD report, Boston University, 1999.

[TC94] Y. Tzeng, K. Chen, W. Kao, A. Fung, A Dynamic Learning Neural Network for Remote Sensing Applications, *IEEE Transactions on Geoscience and Remote Sensing*, 32, 1096-1103, 1994.

[WF94] G. Wilkinson, F. Fierens, I. Kanellopoulos, Integration of Neural and Statistical Approaches in Spatial Data Classification, *Geographical Systems*, Vol. 1, pp. 1-19, 1994.

[YO94] T. Yoshida, and S. Omatu, Neural Network Applications to Landcover Mapping, *IEEE Transactions on Geoscience and Remote Sensing*, 32(5), 1103-1109, 1994.

[ZL65] L. Zadeh, Fuzzy Sets, *Information and Control*, 8, 338-353, 1965.

Chapter 13

REAL TIME FEATURE EXTRACTION FOR THE ANALYSIS OF TURBULENT FLOWS

I. Marusic, G.V. Candler, V. Interrante, P.K. Subbareddy, and A. Moss

Abstract The study of fluid flow turbulence has been an active area of research for over 100 years, mainly because of its technological importance to a vast number of applications. In recent times with the advent of supercomputers and new experimental imaging techniques, terabyte scale data sets are being generated, and hence storage as well as analysis of this data has become a major issue. In this chapter we outline a new approach to tackling these data-sets which relies on selective data storage based on real-time feature extraction and utilizing data mining tools to aid in the discovery and analysis of the data. Visualization results are presented which highlight the type and number of spatially and temporally evolving coherent features that can be extracted from the data sets as well as other high level features.

Keywords: Turbulence, Data Mining, Coherent Vortex Structures, Feature Identification.

1. Introduction

Turbulence is often referred to as the last great unsolved problem in classical physics, and all efforts to develop models to predict turbulent motion have fallen woefully short. The difficulty comes in because turbulent motion is a result of non-linear spatial and temporal interactions across a huge range of length and time scales.

Turbulent flows have great practical importance because they are found in a vast number of engineering applications. Examples include flow over aircraft, spacecraft, and other transport vehicles, flow inside of engines and power plant combustion systems, flow in chemical and waste processing streams, and the flow of blood in the heart and large blood vessels. In these examples, huge cost savings can be achieved if quantities like drag, combustion efficiency, and pollutant output can be predicted accurately for prototype designs. However, these

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 223–238.
© 2001 *Kluwer Academic Publishers.*

quantities cannot be predicted because we have an inadequate understanding of how turbulent *structures* evolve and interact with one another.

Over the last few decades, a broad consensus has emerged that coherent or organized vortical structures (eddies, whorls, or regions of swirling flow) dominate turbulent flows and are the most important physical mechanisms for generating and sustaining turbulent motion. These structures appear in a wide variety of shapes and sizes.

The large scale eddying motions can be seen in nature, as for example, in the flow of water around obstacles, or behind your spoon when you are stirring your coffee. The smallest eddies in the flow are believed to function as sinks of energy, dissipating kinetic energy into heat through fluid friction (or viscosity). The larger scales interact with one another to spawn new larger or smaller scale eddies that cause mixing between adjacent regions of the flow.

To illustrate this interaction, consider the Earth's atmosphere where the largest scale turbulent structures are as large as the height of the clouds or the diameter of a hurricane, while the smallest eddies are about as large as the thickness of a pin. As we know from the variations in the wind strength and direction during an intense storm, the structures at various scales are not static. The structures could be breaking up, dissipating into smaller and smaller ones or they could be coalescing, forming larger scales[1]. These structures are the essence of turbulence, and lacking the information to form a theoretical analysis of their formation and evolution, we are left with having to actually take a deep long look at them in circumstances we believe we control. This can be done with high-quality wind-tunnel measurements or using direct numerical simulations (DNS). These simulations must resolve all possible length and time scales, which results in extremely large calculations, involving the solution of the governing equations on multi-million point grids over many thousands of time steps. Typical calculations require a month or more on the largest parallel supercomputers and are currently limited to simplified flows at unrealistic flow conditions. (Imagine trying to resolve the motion of a large storm that has an overall dimension covering a substantial portion of a continent, while its smallest scale motion is the diameter of a pin!)

Because direct numerical simulations are very costly, an important goal of fluid dynamics researchers is to develop accurate, physically-based numerical models to predict turbulent flows. This model development depends on data from experimental measurements and numerical simulations. With the ever

---

[1] The whole process is neatly paraphrased in the poem by Augustus DeMorgan:

> Great fleas have little fleas upon their backs to bite'em,
> And little fleas have lesser fleas, and so ad infinitum.
> And great fleas themselves, in turn, have greater fleas to go on;
> While these again have greater still, and greater still, and so on.

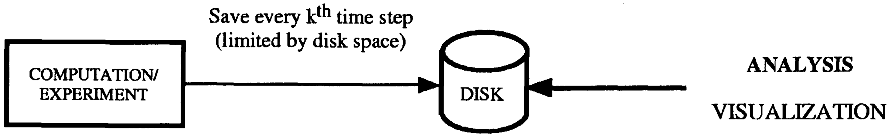


Figure 1. Schematic of a conventional data analysis approach.

increasing speed of computers and the developments in advanced experimental techniques, these methods generate vast amounts of three and four dimensional data, and hence storage as well as analysis of these data has become a major issue and limitation to the extraction of relevant information.

A typical direct numerical simulation of a turbulent flow produces many terabytes of data, of which perhaps only 0.1% is stored and analyzed. For example in the turbulence simulations such as that of Martin & Candler[MC00], data are generated at a rate of approximately 3 terabytes per day, every day for about a month of computing. Typically, only every one-thousandth time step is actually written to disk, with the final database being approximately 2 terabytes. In particle image velocimetry (PIV) experiments data are acquired at a rate of tens of gigabytes per hour, but management and processing of an hour's worth of data typically takes many days. The data-overload situation will only worsen with further advances in computing power and experimental data acquisition hardware. Therefore, the current paradigm in scientific data acquisition and analysis, as represented in Figure 1, has become out-moded.

In this approach, existing visualization tools are used to take a series of averages in an attempt to uncover some type of global information. More recently movies are made of a particular flow variable (typically a scalar or vector quantity) to reach a better, though essentially qualitative, understanding of the flow physics.

Furthermore, data are stored periodically as the simulation progresses; however the interval between storage is not a phenomenological one and is usually assigned at the start of the simulation without regard to the flow evolution. Storing data at regular intervals can only capture instantaneous snap-shots of the flow and has little hope of capturing relevant time and space evolving dynamics. This is not to say that researchers are performing studies incorrectly. Rather there is presently no means of efficiently storing and analyzing the enormous amounts of data being generated. For example, the wall turbulence simulation data of Spalart(1988) is *still being analyzed thirteen years after it was*

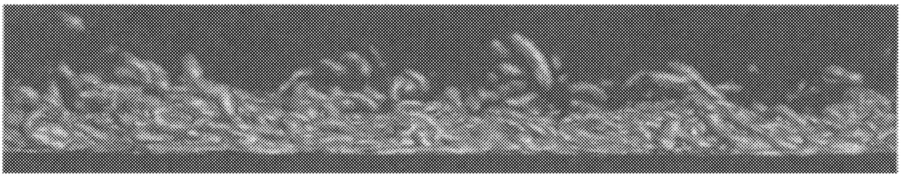


Figure 2. Coherent structures in a turbulent boundary layer flow extracted using discriminant-based structure identification. (See section 3 for details)

generated. Several researchers are still using this data set to provide a description of wall turbulence and developing a better *feel* for turbulent fluid motion [R91, CCB97, CC00]. Figure 2 shows a typical result from such studies where vortical structures are identified in a turbulent channel flow. Such techniques have been very useful in helping to identify what structural features exist in the flow. However, without improved data analysis methods, advances in computing and data acquisition technology will not be accompanied by an improved understanding of the underlying physical processes.

In this article we wish to outline a new approach to the analysis of large scientific data sets involving dynamic and transient processes. Successful implementation will involve developing new interactive visualization and feature extraction tools and employing newly-developed scalable data mining methods.

2. New Approach for Data Analysis

Conventional data analysis and visualization separates the analysis of the data from its generation. This is inadequate for the analysis of large scientific data sets. There is simply too much data, it takes too long to make "pictures" of it, and there is too much to look at all at once. Even if it were somehow possible to save and look at everything, we might not necessarily want to. It can be argued that some of the most significant advances in understanding will come, not from showing more, but from showing less. Instead of relying on faster computers to visualize the data we advocate a new methodology, as represented in Figure 3.

The approach involves monitoring the simulation or experiment and selectively storing the data in close proximity to an event of interest – termed the "trigger." For example, while a wind tunnel is being run, the data would be streamed to a multi-processor computer where it is stored temporarily and analyzed in real time. As more data are generated, the oldest data are thrown away until an event of interest is detected through application of advanced visualization and feature recognition techniques. Then the data remaining in memory in the spatial and temporal vicinity of the event are identified as relevant and are

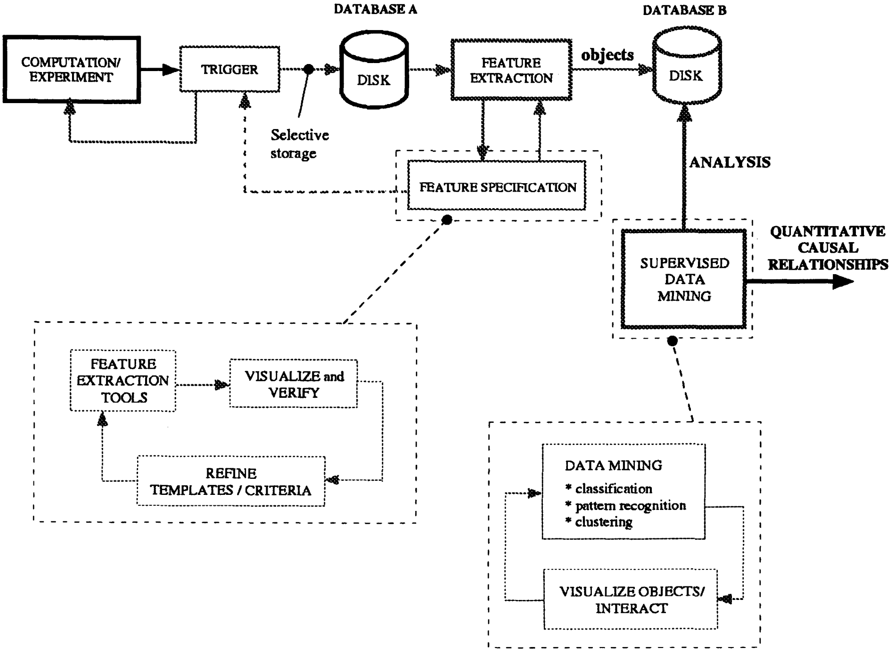

DATABASE A
DATABASE B
COMPUTATION/ EXPERIMENT
TRIGGER
DISK
FEATURE EXTRACTION
objects
DISK
Selective storage
FEATURE SPECIFICATION
ANALYSIS
SUPERVISED DATA MINING
QUANTITATIVE CAUSAL RELATIONSHIPS
FEATURE EXTRACTION TOOLS
VISUALIZE and VERIFY
REFINE TEMPLATES / CRITERIA
DATA MINING
* classification
* pattern recognition
* clustering
VISUALIZE OBJECTS/ INTERACT

Figure 3. Schematic of our new data analysis approach. The connected boxes enclosed by dotted lines are essentially the same, with the larger boxes showing the internal features of the smaller ones, and have been drawn separately for clarity.

written to disk (Database A). Specifying the trigger criteria requires a testing period wherein the correctness of the criteria are determined by a trial and error process. Correct trigger identification is critical since using a faulty trigger would result in the loss of valuable data. A similar approach would be used to detect relevant information in a numerical simulation. In this case, data are analyzed as generated and trigger events are identified on the fly. This is in contrast to the existing paradigm in which data are written to disk at arbitrary intervals at a rate predetermined by the available disk storage (see Figure 1). Using visualization and feature extraction to generate Database A is, in itself, a major challenge in the analysis of these flows.

With all the relevant data stored to disk, feature extraction and visualization techniques are then used to further analyze the data. The feature specification criteria need to be formulated to identify and extract "objects" of interest from the database. An object quantifies the key features of each relevant turbulent structure, such as its strength, spatial extent, persistence, and location in space-time. This is where strong collaboration and interaction between the engineering and computer science/visualization experts is needed. An iterative process is required, in which the extraction criteria are refined to identify the features relating to the event. Once the module has been trained, the feature extraction can be run to produce a new compact database of objects along with their parametrizations.

This new database of objects (Database B) is then analyzed by using data mining techniques to search for patterns of interest and provide high-level information about the objects, and thus help unravel the physical mechanisms and causal relationships in the flow. Again, this requires close interactions between the engineering and computer science experts. Successful implementation of these tools would result in a powerful new means to visualize and interpret mechanisms which are currently elusive. It will allow us to pose an entirely new class of questions concerning how one event causes another in turbulent flows.

3. Case study: a Mach 4 turbulent boundary layer

Let us consider a specific turbulent flow, which is illustrated in Figure 4: the turbulent boundary layer created on the Space Shuttle during re-entry into the atmosphere. The turbulence is generated in the narrow region between the surface of the Shuttle and the high-speed external flow. In this boundary layer, there are extreme velocity and temperature gradients that cause the flow to become turbulent. The turbulent motion affects the aerodynamic forces acting on the Shuttle, but more importantly for this application, the turbulence significantly increases the aerodynamic heating of the Shuttle.

Boundary layer flows also determine the lift and drag performance of aerodynamic surfaces such as wings, propellers, and turbine blades. The turbulent motion determines if a flow will separate, which results in sudden loss of lift and substantial drag losses. A large portion of the engine power of aircraft, trains, and automobiles goes into overcoming turbulent drag losses. Boundary layers also occur on combustor surfaces and they are often the site of combustion inefficiencies and quenching that result in unwanted pollutant formation.

Visualization studies have provided a good qualitative appreciation of the structures that exist in wall turbulence and have changed the traditional notion that turbulence is a completely random phenomenon. As discussed above, it is now clear that turbulent boundary layers are largely made up of organized and quasi-deterministic, coherent vortex structures; an example is seen in Figure 2. Physical models for the kinematic state of the flow have been proposed based on these ideas using a statistical distribution of vortex structures [PM95, MP95]. These formulations have been largely successful in reproducing second-order statistics of the turbulence, given the mean flow distribution. Unfortunately, being kinematic stochastic models, they say nothing about the dynamic interaction between the turbulent structures, and we still need to understand how

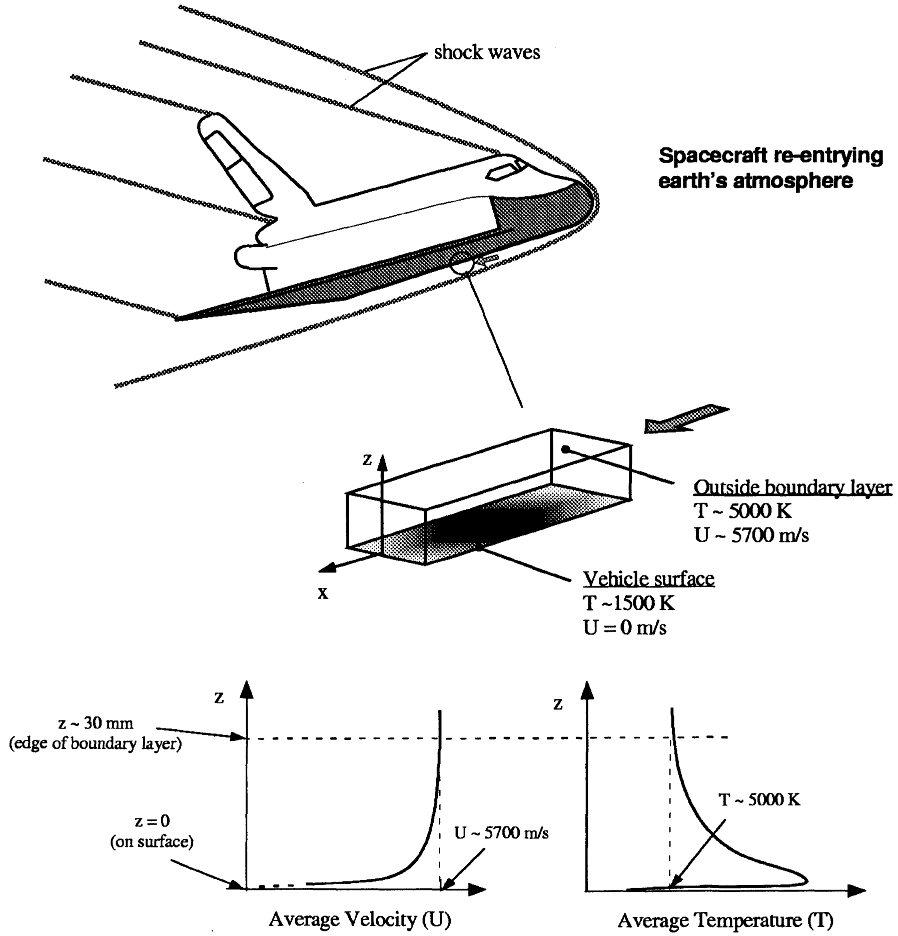

shock waves
Spacecraft re-entrying earth's atmosphere
z
x
Outside boundary layer
T ~ 5000 K
U ~ 5700 m/s
Vehicle surface
T ~1500 K
U = 0 m/s
z
z ~ 30 mm
(edge of boundary layer)
z = 0
(on surface)
U ~ 5700 m/s
Average Velocity (U)
z
T ~ 5000 K
Average Temperature (T)

Figure 4. Illustration of a turbulent boundary layer flow on the Space Shuttle during re-entry. Because of the huge computational cost, only a small portion of the flow can be simulated; this is denoted by the boxed region. The plots in the lower half of the figure illustrate the average velocity (on the left) and average temperature as a function of the distance from the wall, z.

these structures are generated and evolve. Thus, information on the dynamic interaction between between structures would greatly improve our understanding of turbulence.

Of particular practical interest is the phenomenon known as "bursting", which is characterized by a violent ejection of near-wall fluid into the outer part of the boundary layer, resulting in large drag and heat transfer. The frequency of bursting is directly related to the skin friction drag [LB98] and if we could devise schemes that reduce the average time between bursts, significant benefits

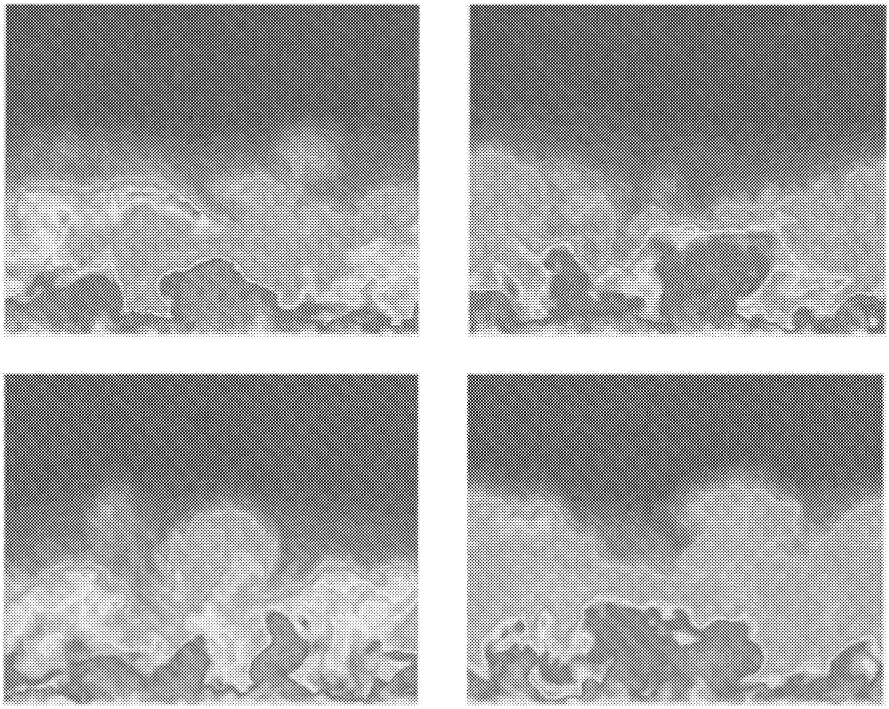


Figure 5. Example of "bursting" shown in spanwise planes from the Mach 4.0 adiabatic wall boundary layer simulation of Martin & Candler [MC00].

would result. For example, for every 1% reduction in skin friction achieved, the US aviation industry would save an estimated $200 million per year in fuel consumption. An example of the bursting phenomenon was captured in the high Mach number boundary layer simulation by Martin & Candler [MC00] and is shown in Figure 5. Here representative spanwise planes (the flow is into the page) are shown at one time step and temperature is the variable plotted. We see large coherent structures that lift the high temperature near-wall fluid (white and red in Figure 5) into the cool free-stream (blue). Apart from large drag, these bursting events are responsible for increasing the heat transfer rate by a factor of about four compared to a non-turbulent laminar boundary layer. If we could understand how the events are generated, it may be possible to develop actuators to suppress or stimulate their formation, depending on whether decreased or increased heat transfer is desired.

3.1. Details of simulation

The flow conditions for this case study are characteristic of a turbulent supersonic boundary layer at high altitude flight conditions. The data are from a large direct numerical simulation (DNS) data set of Martin & Candler [MC00] for a Mach 4 turbulent boundary layer with an edge temperature of 5000 K and density of 0.5 $\mathrm{kg/m^3}$. These conditions are representative of the flow sketched in Figure 4.

The DNS solves the discretized time-dependent compressible Navier-Stokes equations, which essentially represent the conservation of mass, momentum, and energy. This was done using a high-accuracy, low-dissipation, shock-capturing finite difference method specifically designed for this type of problem [WC97]. The time integration uses a parallelizable second-order accurate implicit method [OC97].

The grid in the streamwise and spanwise directions is equispaced and the wall-normal direction is geometrically stretched from the wall. The domain size is measured in units of the boundary layer thickness δ, and for this case is $8\delta \times 2\delta \times 15\delta$ [MC00]. The grid resolution is $\Delta x^+ = 15$, where the superscript denotes normalization using the viscous length scale (ν/U_τ), $\Delta y^+ = 11$, with Δz^+ varying from 0.15 to 46 across the boundary layer. To achieve adequate resolution for this domain size, the DNS was performed on a grid with $384 \times 128 \times 128$ points in the streamwise, spanwise, and wall normal directions, respectively.

The calculations were performed using a 256 processor Cray T3E at the University of Minnesota Army High Performance Computing Research Center (AHPCRC). There are no detailed experimental data for these flow conditions, but the results of the DNS agree very well with theoretical and experimental correlations. It should be noted that the DNS results are statistically exact and any variable of interest can be constructed from the solution at any given time.

3.2. Identification of structures

The structures that form the basis of turbulent motion are believed to be composed of regions of high vorticity, or angular velocity, which very roughly speaking means that the fluid particles are spinning around themselves and other fluid particles. The picture is confused however, by the fact that there are regions of flow with high vorticity, but are not part of what are believed to be coherent structures.

In order to separate the "spurious" vorticity from that within the structures of interest, we must consider an appropriate frame of reference. For example, the wake behind a boat appears to be highly disorganized when seen from the shore, but if one is on the boat, the wake assumes an organized structure. Similarly, if we move with a particle of fluid in the flow, it becomes clear whether the fluid is swirling about that point and is part of a coherent vortical structure.

It is also useful to consider the local streamline pattern at a point, which is computed using the gradients of each of the components of the velocity. This velocity gradient tensor ($\partial u_i/\partial x_j$, where i and j run over all three of the spatial dimensions) contains all nine possible velocity derivatives. Visualizing a field made up of nine quantities at each point is usually a futile exercise, and we must distill one number out of this matrix that quantifies the swirling vortical motion. Quite naturally, we are led to the eigenvalue problem for the matrix. Being a 3×3 matrix, it has 3 eigenvalues which are either all real or have a pair of complex roots and one real root. A little analysis shows that the latter case is when the local streamlines exhibit the swirling pattern that we were looking for, and in fact, the complex part of the eigenvalues indicates the strength of the swirl. The number that determines whether there are real or complex roots is the discriminant (D) of the characteristic equation of the eigenvalue problem. If $D > 0$, we have a swirling flow, and a connected region of positive values of D would be an unambiguous definition for a coherent structure. It is this parameter, D, that we currently use to identify the structures in the flow.

3.3. Visualization tools and results

The DNS data are rendered using a classical ray-casting volume renderer (Levoy [L88]). At each time step, the 3D normalized level-set data are first resampled into a high resolution uniform grid (1018 × 251 × 137), and then to minimize disk space and memory requirements quantized into 8-bit values spanning the range 0-255, such that the remapping is uniform across all time steps. Images are created by tracing a bundle of rays from the eye point through the 3D data, such that each ray passes through the center of a pixel in the image plane, and returns the color value to be assigned to that pixel. The final pixel color is obtained by compositing local color and opacity values trilinearly interpolated from the uniform grid to sample points at evenly spaced intervals along the ray.

Local color values are computed at each grid point using a simple two-sided Phong illumination model (Foley *et al.* [FDFH90]):

$$\text{sample\_color} = k_a + k_d \times |N \cdot L| + k_s \times |N \cdot H|^n,$$

where k_a, k_d and k_s are the ambient, diffuse and specular reflectivities multiplied by the intensity of illuminant, respectively; which together with the exponent n are defined by the user. N is a unit vector in the direction of the data value gradient, which can be interpreted as the unit normal to the level surface at the grid point, L is the direction of the parallel illumination from an infinitely distant light source, and H is the vector halfway between L and the line of sight, or direction of projection. Local opacity values are defined at each grid point as a gradient magnitude-weighted linear function of the input data values according to a user-specified data-to-opacity correspondence given at key breakpoints. The volume rendering approach in effect finesses the definition of the apparent surfaces, allowing greater flexibility to portray the structures of interest in a form that is robust to the local effects of isolated samples and minor fluctuations or nonuniformities in the distribution of the chosen level-set. Our experience shows that this volume rendering yields more clearly connected structures than the usual iso-surface rendering. Examples of a volume rendering using the discriminant, D, to specify the illuminant for one time step from the DNS are shown in Figures 2 and 6.

In order to isolate a particular structure from the forest of other structures in the flow field, we use a volume filling method. A seed is placed within a structure of interest that is usually identified from an image such as shown in Figures 6 and 7. Then a nearest-neighbor search is done to find all connected voxels where the variable of interest (D here) is greater than a user-specified value. This connected region is then visualized as a colored rendering. A more sophisticated volume filling method that uses gradient information may result in more consistent connectivity of structures; this extension is being pursued in continuing work.

The volume filling approach can be used to construct a database of relevant objects for data mining analysis. For example, it would be straight-forward to identify all locations within a flow domain where the discriminant is above a certain value; these locations are used as seeds for the volume filling. The structures are then filled and any duplicate structures are eliminated. This results in a finite number of connected regions; the number of which depends on the specified seed and connectivity thresholds. Each structure can be parameterized by its location, size, volume-averaged swirl strength, or many other possible variables.

We would like to determine how these structures evolve in time. Figure 7 shows a typical evolution of one large-sized volume-filled structure over approximately 90 time-steps in the simulation. Since the structures move long with the flow, we can estimate to high accuracy where they will be located at the next time step. The volume filling can then be repeated, but with a very good estimate of the location and extent of the connected regions. Thus, once we have performed an initial volume filling and structure identification, updating the database of structure connectivity and parameters is straight-forward. The only concern with this approach is that entirely new structures can be born, or two structures can merge. Appropriate logic must be included to allow these processes to occur. However from our current visualizations, the frequency of large structure birth and merging appears to be quite low (this is not true for the smaller structures).

In any case, the volume filling can be used to construct a database of objects of interest. The database will contain a space-time trajectory of each structure, including all relevant parameters. Because each entry in the database represents a large ensemble of relevant voxels, it will be very significantly smaller than the raw data. For example, a representative number of large-scale structures

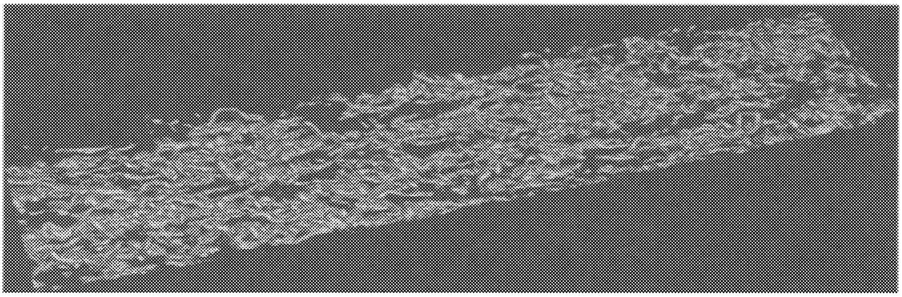


Figure 6. An isometric view of the boundary layer from the Mach 4.0 adiabatic wall simulation.

may be several hundred; if each structure is parameterized by 10 quantities, a single time step will require only on the order of 10^4 real variables per time step. This is contrasted with the 3×10^7 real variables required to describe the entire domain. This represents a compression of a factor of 10^3; but more importantly, the small database contains only the *relevant* data.

4. Data Mining Issues

A primary issue with mining of the turbulent flow data set is the interface between the fluid dynamicists and the data miners. Fluids investigators would like to be able to ask quite abstract questions, such as "What causes a burst event?" or "Do structures group themselves together, or are they random?" These questions need to be converted into a form that is appropriate for data mining, and of course the compact database discussed above must contain the relevant information to answer a given question. Likewise, it is important for the data mining investigators to make clear what classes of questions data mining algorithms can actually answer. Thus, once we have crossed the hurdle of generating a compact data set, probably the main issue is a matter of communication between the fluid dynamicists and the data miners.

Another issue is that the turbulence is extremely complicated with a very large number of vortical structures across a very large range of length scales, as seen in Figure 6. This complexity in the flow field results in having to consider many possible mechanisms to explain a given event or process. For example, consider the problem of understanding burst formation as indicated in Figure 5. It is expected that to predict burst formation in boundary layer turbulent flow, a model will involve both physical properties (temperature, pressure, vorticity, etc.), as well as spatial and temporal relations between the different vortical structures in the flow. Unfortunately, with our present understanding of turbulence, we do not know what these relations may be, and there is a very large number of possible correlations between these variables and their history. For

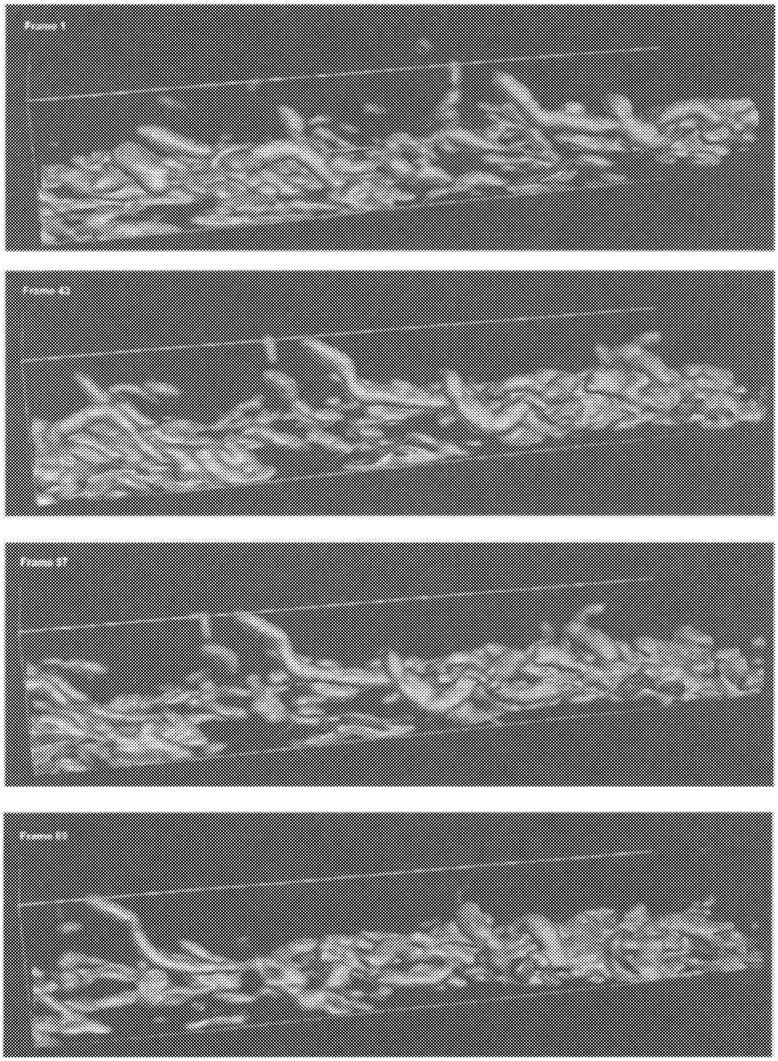


Figure 7. Figures above show the time evolution of a typical structure seen in the simulation.

example there are a number of higher level characteristics which are speculated to cause burst formation. These include the organization of vortices into coherent spatial clusters and spatial coherence among vortices or clusters of vortices [AMT00, ZABK99, M01]. Thus, models discovered by data mining techniques need to be able to express such relations.

As an initial approach, *training sets* need to be established. In a steady-state turbulent flow, bursts occur at unpredictable times during a simulation. Using the intelligent data filtering techniques discussed above, we can obtain objects describing the state of the flow at a time interval around the occurrence of each burst. These observations will become the *positive* instances in our training set. At the same time, we can construct a large number of objects describing the state of the flow during time intervals in which no bursts occur; these become the *negative* instances in our training set. From this training set, we can begin developing explanatory models that will yield quantitative insight into burst formation. These models can then be evaluated using simulation data that has not been used in the training of the model.

Unfortunately, even though the feature extraction techniques can provide us with higher order objects rather than just raw flow information, the information content of these objects is still too low to allow us to express the higher order relations that are needed. Thus, the need exists for data mining techniques based on clustering and pattern discovery that will analyze the low level objects and extract higher order features present in the data set that will aid in the development of explanatory models for the phenomena of interest.

There are several characteristics of the objects extracted from scientific data sets that makes clustering and pattern discovery challenging. Irrespective of the sophistication of the feature extraction tools, the objects extracted from the raw flow information will unavoidably have a certain degree of noise and imprecision. Similarly, the number of objects in the database can easily run to several thousands and increases as the simulation/experiment duration increases. At the same time, the parametric description of these objects can easily involve tens of features, and since the objects arise in a three dimensional time varying simulation, they have spatial as well as temporal features. Existing data mining techniques do not appear to be well suited to operate on data sets that combine the above characteristics.

Developing clustering algorithms for scientific data sets poses a number of challenges due to both the characteristics of the data itself as well as the types of the desired clusters. The clusters of objects may be of variable sizes and densities, and may be of arbitrary shapes. The need exists to develop clustering algorithms that are capable of accommodating different clustering objectives. For example, the problem of finding a cluster that is a packet of vortices is entirely different than that of finding a cluster that contains vortices of the same size.

In the context of pattern discovery algorithms, the patterns to be considered here involve spatial, temporal, or spatio-temporal relations among objects. For example, interesting patterns may involve vortices oriented in a certain way, or a certain type of vortices formed after another type, or a set of small vortices evolving to create larger vortices.

The problem of finding these patterns becomes even more challenging when the objects are higher order features such as clusters. In this case establishing relationship among clusters from different events itself can be non-trivial. The need therefore exists to extend existing pattern discovery algorithms by allowing them to discover arbitrarily complex patterns that represent the relationships among events.

Some preliminary results of Data Mining in turbulent flows are discussed in the accompanying chapter by Han,Karypis and Kumar.

Acknowledgments

We would like to thank Vipin Kumar, George Karypis and Eui-Hong (Sam) Han for sharing with us their insights into Data Mining. This work is supported by the support of the National Science Foundation through grant ACI-9982274, and this is gratefully acknowledged. This work was also supported in part by the U.S.Army High Performance.Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement DAAH04-95-2-0003 / contract DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Computer time was provided in part by the Minnesota Supercomputing Institute.

References

[MC00] M.P. Martin and G.V. Candler. DNS of a Mach 4 boundary layer with chemical reactions. *AIAA-2000-0399*, 2000.

[R91] S.K. Robinson. Coherent motions in turbulent boundary layers. *Annu. Rev. Fluid Mech.*, 23:601-639, 1991.

[CCB97] B.J. Cantwell, J.M. Chacin, P. Bradshaw. On the dynamics of turbulent boundary layers. In *Self-Sustaining Mechanisms of Wall Turbulence,* Ed. R.L. Panton, Comp. Mech. Publications, 1997.

[CC00] J.M. Chacin and B.J. Cantwell. Dynamics of a low Reynolds number turbulent boundary layer. *J. Fluid Mech.,*404:87-115, 2000.

[PM95] A.E. Perry and I. Marusic. A wall-wake model for the turbulence structure of boundary layers. Part 1. Extension of the attached eddy hypothesis. *J. Fluid Mech.,* 298:361-388, 1995.

[MP95] I. Marusic and A.E. Perry . A wall-wake model for the turbulence structure of boundary layers. Part 2. Further experimental support. *J. Fluid Mech.,* 298:389-407, 1995.

[LB98] J.L. Lumley and P. Blossey. Control of turbulence. *Annu. Rev. Fluid Mech.,*30:311-327, 1998.

[WC97] V.G. Weirs and G.V. Candler. Optimization of weighted ENO schemes for DNS of compressible turbulence. *AIAA-97-1940,* 1997.

[OC97] D. Olejniczak and G.V. Candler. Numerical testing of a data-parallel LU relaxation method for compressible DNS. *AIAA-97-2133*, 1997.

[L88] M. Levoy. Display of surfaces from volume data. *IEEE Comp. Graphics App.*, 8(3):29-37, 1988.

[FDFH90] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics, Principles and Practice.* 2nd ed., Addison Wesley, 1990.

[AMT00] R.J. Adrian, C.D. Meinhart and C.D. Tomkins . Vortex organization in the outer region of the turbulent boundary layer. *J. Fluid Mech.*, 422:1-53, 2000.

[ZABK99] J. Zhou, R.J. Adrian, S. Balachandar and T.M. Kendall. Mechanisms for generating coherent packets of hairpin vortices in channel flow. *J. Fluid Mech.*, 387:353-396, 1999.

[M01] I. Marusic. On the role of large scale structures in wall turbulence. *Phys. Fluids* 13:735-743, 2001.

Chapter 14

DATA MINING FOR TURBULENT FLOWS

Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar

Abstract Data mining techniques hold great promise for enabling the automatic analysis of large data sets generated by scientific simulation, and thus, may help engineers and scientists unravel the causal relationships in the underlying system. In this chapter, we propose several data modeling methods to incorporate spatial and temporal features of scientific simulation data and investigate some of them in the context of developing models for predicting burst events in turbulent flow. We use the classification rules algorithm C4.5rules and support-vector machines on the turbulent flow simulation data to develop predictive models for identifying upward or downward velocity movements of the flow close to the wall as a function of swirl strength in the nearby region.

Keywords: data mining, turbulent flows, classification rules, support vector machines

1. Introduction

Turbulent flows are found in a vast number of engineering situations such as the flow over aircraft, the flow inside of engines and power plant combustion systems, etc. As discussed in the previous chapter [MCI$^+$01], huge cost savings can be achieved if quantities such as drag and combustion efficiency can be predicted accurately for prototype designs. However, the direct prediction of these quantities is not feasible even in relatively simple geometries, because solving the equations describing these flows inflicts a massive cost in terms of computational resources. To be able to make such predictions, scientists need to develop a greater understanding of how turbulent structures evolve and interact with one another. They need to find answers to questions such as "What causes a drag-producing "burst" event?", "Do vortical structures group themselves together, or are they random?", "What are the correlations among variables of physical properties (temperature, pressure, vorticity, etc.)",

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 239–256.
© 2001 *Kluwer Academic Publishers.*

and "What are the spatial and/or temporal relations between the different vortical structures in the flow?" Numerical simulations can potentially provide answers for the above questions. However, the huge amount of data being generated by these computational simulations makes it impossible to perform such an analysis manually.

Data mining techniques can be used to develop a new set of analysis tools that can make it possible for engineers and scientists to gain insight into the underlying mechanisms of dynamic physical processes. Once raw simulation data is converted into a suitable form, traditional data mining techniques, such as clustering, predictive modeling, and pattern discovery can be applied.

Clustering techniques can find sets of objects or events that share similar characteristics, and can be used to answer questions like "Do vortical structures group themselves together?", "Are they random?", or "What are the different kinds of bursting events or vortical structures in the flow?" Pattern discovery algorithms (e.g., association rules algorithms) can be used to determine correlations among physical properties and spatial/temporal features of the turbulent flow structures. A predictive model, in the form of decision trees [Qui93], regression models [BFOS84], Markov models [DK01], etc., allows one to estimate the likelihood of an event of interest in the spatial neighborhood or in the near future, given values of other variables. These models can allow scientists to answer questions like "What causes a burst event?", or "What causes increases in the vorticity?"

In this chapter, we investigate the problem of developing predictive models for scientific simulation data. Specifically, we present preliminary results on the development of a model for predicting upward or downward velocity movements of the flow close to the wall as a function of the swirl strength ([MCI$^+$01] of nearby regions. These wall-normal movements are a potential cause of drag, and are associated with burst events where large amounts of near-wall low-momentum fluid are ejected into the outer higher momentum region of the boundary layer.

2. Data Modeling Issues

Data mining algorithms for building predictive models [Qui93, Coh95, HKK01, CV95, Joa98, DMT94] require a training set that consists of a collection of record each having a number of attributes. One of the attributes is the target attribute, also referred to as the dependent attribute, while other attributes are referred to as independent attributes. These algorithms build a model that predicts the value of the target attribute as a function of the independent attributes.

Table 1. A possible set of records generated from turbulent flow simulation data.

(x,y,z)	time step	swirl strength	velocity movement direction/strength
(1,1,1)	1	3.0	-2.0
(1,1,2)	1	2.5	-2.5
(1,1,3)	1	2.0	-1.0
(1,1,4)	1	1.0	0.2
⋮	⋮	⋮	⋮
(1,1,1)	2	1.0	-0.2
(1,1,2)	2	2.5	-0.1
(1,1,3)	2	1.0	0.0
(1,1,4)	2	1.0	0.1
⋮	⋮	⋮	⋮
(100,100,97)	50	3.5	2.5
(100,100,98)	50	3.5	2.5
(100,100,99)	50	3.5	2.5
(100,100,100)	50	4.5	1.5

In scientific simulation, the data usually consists of a sequence of frames corresponding to different time steps in the simulation. Each frame is a 3D grid of points, and there is a vector of variables associated with each grid point. The objective is to develop a model that can predict the value of one of the variables for a grid point as a function of the other variables. Conventional data mining algorithms can be used to develop such predictive models only after a training set of records is prepared from the sequence of 3D frames.

Given raw simulation data, many different possibilities exist for preparing a set of records. A simple method is to associate a record with each grid point of a frame, and make each of the associated variables an attribute in the record. Table 1 illustrates this for a turbulent flow simulation data in which two variables (swirl strength and velocity movement direction/strength) are associated with each grid point. Each row of the table corresponds to a record. This method is quite restrictive, as it allows the velocity movement direction/strength to be modeled only as a function of swirl strength at the same grid point.

To allow bursting events to be modeled as a function of the swirl strength of nearby grid points, we can extend the record for each grid point by adding additional attributes that are derived from the variables of the neighboring grid points. Many choices are possible for the neighborhood as well as for deriving attributes from neighboring cells. Neighborhoods can be volumes of arbitrary sizes and shapes. An attribute can be a raw variable at the grid points in the neighborhood, or it could be derived from these raw variables (e.g., average value over a set of grid points). Figure 1 shows two possible methods for deriving attributes based upon neighborhood cells.

The value of a variable at a grid point may also be influenced by the value of the variables at the same grid point (or in the neighborhood)

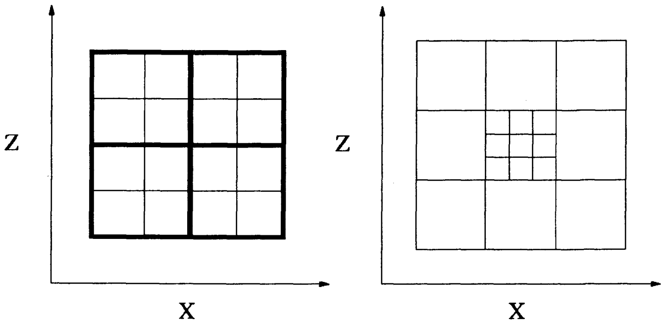


Figure 1. Two possible methods for deriving attributes based upon neighborhood cells.

at previous time steps. To incorporate this temporal dimension, records can be defined based on a sliding time window w. For example, given $w = 3$, each row under time steps t_0, t_1, and t_2 of Table 2 will form a transaction, each row under t_1, t_2, and t_3 will form a transaction, each row under t_2, t_3, and t_4 will form a transaction, and so forth.

More complex attributes can be obtained from domain specific high-level structures such as hairpin vortices in turbulent flow simulation. Presence or absence of these high level structures themselves can serve as useful attributes. Parameters or characteristics of these high level structures can also be used as attributes. For example, a hairpin vortex may have parameters like width of the vortex core, the vertical distance from the vortex head to the solid surface, and the angle that the vortex makes with the surface. Interesting events at a grid point (or a region) such as "temperature above 50% of the average temperature," "10 consecutive time steps with increasing pressure," "3 standard deviations above the mean value of vorticity," "emergence or disappearance of a hairpin vortex", or "above 75 degree angle between the hairpin vortex and the surface", can also serve as attributes of transactions.

In summary, there are many possible methods for defining records for scientific simulation data. The success of the followup data mining algorithms depends critically upon the method chosen for generating records. Hence, close cooperation between domain scientists and data mining researchers is critical for proper definition of transactions and attributes.

Table 2. Combined swirl and velocity data based on XY plane at Z=τ with neighborhood box ($N_x \times N_y \times N_z$) of 2D grid points.

2D Grid	t_0						t_1						t_2						⋯
	swirl in neighbor 3D grids					velocity	swirl in neighbor 3D grids					velocity	swirl in neighbor 3D grids					velocity	⋯ ⋯
(1,1)	1	0	1	⋯	0	1	1	0	0	⋯	0	1	1	0	0	⋯	0	0	⋯
(1,2)	0	0	1	⋯	0	0	1	0	0	⋯	0	0	0	1	0	⋯	0	0	⋯
(1,3)	0	1	0	⋯	0	0	0	0	1	⋯	0	-1	1	0	0	⋯	1	0	⋯
(1,4)	0	1	0	⋯	0	0	0	1	0	⋯	0	0	0	1	0	⋯	0	0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
(2,1)	0	0	0	⋯	1	1	0	0	0	⋯	0	1	1	0	1	⋯	0	0	⋯
(2,2)	0	1	0	⋯	0	0	0	0	1	⋯	0	-1	0	0	0	⋯	1	-1	⋯
(2,3)	0	0	0	⋯	0	0	0	1	0	⋯	0	0	1	1	0	⋯	0	1	⋯
(2,4)	0	0	1	⋯	0	0	1	1	1	⋯	0	0	1	0	1	⋯	0	0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

3. Data Modeling for Predicting the Burst Event

We present an approach to predict/explain the "bursting" event, which is the violent upward or downward movement of near-wall fluid, in terms of swirl structures. At each time step of the simulation, a high resolution uniform grid ($i \times j \times k$) is formed over the simulation space. For each grid point, we have a measurement of the swirl strength in terms of the discriminant (D) of the characteristic equation of the eigenvalue problem described in [MCI$^+$01]. A grid point has value one if D is greater than swirl threshold (s_{thresh}) and zero otherwise.

At each time step, the XY plane at $Z = \tau$ in the direction normal to the wall (i.e., Z direction in XYZ coordinate) is chosen as the plane to measure the "bursting" event, or upward or downward velocity movement. Each grid point on the XY plane has value 1 if the upward velocity is greater than the velocity threshold v_{thresh}, -1 if the downward velocity is greater than v_{thresh}, and 0 otherwise.

Table 2 shows the combined swirl and velocity data. Several possibilities exist for defining transactions with the data set represented in Table 2. Given a grid point P of the XY plane at Z=τ, we define a neighborhood box with size $N_x \times N_y \times N_z$ such that P is in the center of the box. Depending on how the neighborhood box is defined, a transaction can contain different information. For example, if the neighborhood box is defined to be 1x1x8, then only the swirl strengths above and below the XY grid point are considered. On the other hand, if the neighborhood box is defined to be 8x8x8, then swirl strengths in a wider spatial area are considered.

The attributes (or variables) in the record can be defined in several ways. Given a neighborhood box of $N_x \times N_y \times N_z$, a simple way of

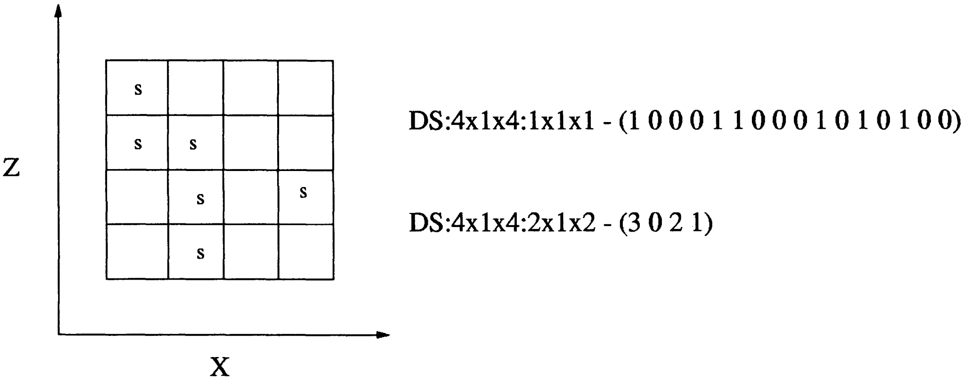

s
s
s
s
s
s
Z
X
DS:4x1x4:1x1x1 - (1 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0)
DS:4x1x4:2x1x2 - (3 0 2 1)

Figure 2. A record with 4x1x4 neighborhood box. Grids with swirl activity are marked with letter ‘s’. Each attribute is shown in the order of top to bottom and left to right.

defining attributes is to associate a binary attribute to each grid point in this neighborhood box. Each such attribute indicates the presence or absence of swirl in the grid point. For example, for a neighborhood box of 8x8x8, there are 512 binary attributes. Figure 2 shows a record with 4x1x4 neighborhood box. In this example, there are 16 binary attributes. Another way of defining attributes is based on smaller boxes in the given neighborhood box. For example, a 8x8x8 neighborhood box contains 64 boxes of size 2x2x2 in it. An attribute can be associated with each such 2x2x2 box. The value of each attribute can be the number of grid points within the 2x2x2 box that have swirl value 1. Figure 2 shows an example record in DS:4x1x4:2x2x2. In the subsequent discussion we denote data set generated from $N_x \times N_y \times N_z$ neighborhood box with $n_x \times n_y \times n_z$ size box inside as DS:$N_x \times N_y \times N_z$:$n_x \times n_y \times n_z$.

Another way of defining attributes is based on shells in the neighborhood box. For example, given a 16x16x16 neighborhood box with 4096 grids, each variable can correspond to the number of grids with swirl value in grids of distance 0-2 from the center, 2-4 from the center, ..., and 10-12 from the center. We call this data set 16x16x16:radius-1. Figure 3 shows a record with 4x1x4 neighborhood box. We can also double the number of variables by dividing each shell of 16x16x16:radius-1 in half using the XY plane that goes through the center of the neighborhood box. Hence, this dataset can have 12 variables, where each variable corresponds to the number of grids with swirl value in grids of distance 0-2 from the center and below the XY plane, distance 0-2 from the center and above XY plane, distance 2-4 from the center and below the XY plane, distance 2-4 from the center and above XY plane, ...,

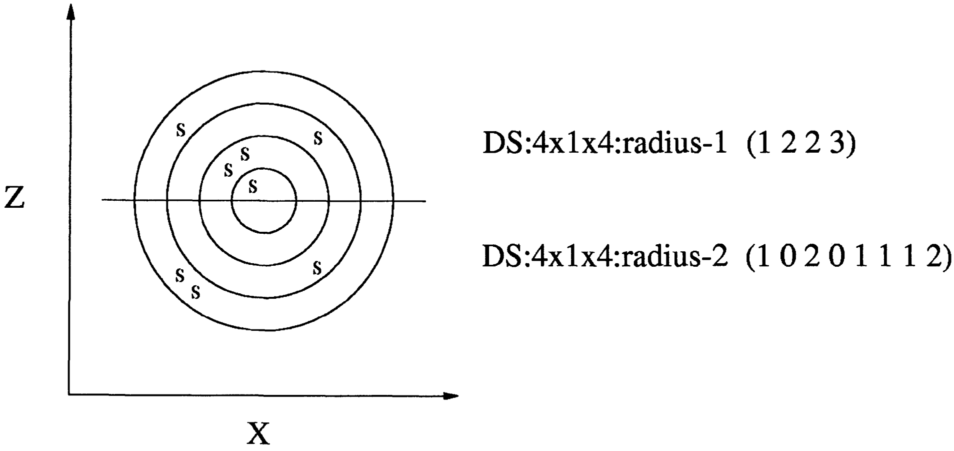

s
s
s
s
s
Z
s
s
s
DS:4x1x4:radius-1 (1 2 2 3)
DS:4x1x4:radius-2 (1 0 2 0 1 1 1 2)
X

Figure 3. A record with 4x1x4 neighborhood box. Grids with swirl activity are marked with letter 's'. Each attribute is shown in the order of inside to outside and top to bottom.

distance 10-12 from the center and below the XY plane, and distance 10-12 from the center and above the XY plane. We call this data set 16x16x16:radius-2. A record in a data set, DS:4x1x4:radius-2 with 8 attributes, is shown in Figure 3. We can also construct another data set, called 16x16x16:radius-4, by dividing each region that was formed to generate variables for DS:16x16x16:radius-1 into 4 by the XY plane and YZ plane that goes through the center of the neighbor box. Similarly, DS:16x16x16:radius-8 can be constructed by dividing each region that was formed to generate variables for DS:16x16x16:radius-1 into 8 by the XY plane, YZ plane, and XZ plane that goes through the center of the neighborhood box.

4. Experimental Results

The data set is a part of a Large Eddy Simulation (LES), which is very similar to the direct numerical simulation (DNS) data set discussed in [MCI$^+$01], the difference being that DNS has a fully resolved grid while LES uses a coarser grid and models the unresolved fine scale motions. Since this was a preliminary investigation, we felt that the smaller LES datasets would be appropriate. This dataset has $48 \times 64 \times 37$ grid points in the streamwise (X axis), spanwise (Y axis), and wall normal directions (Z axis), respectively. The velocity measurement was taken on the XY plane at the grid number 24 of wall normal directions. For this experiment, the swirl threshold (s_{thresh}) was 95% of the maximum swirl of the data and velocity threshold (v_{thresh}) was 150.0 meters per second.

In the experiments we performed, we focused on finding spatial patterns without considering temporal aspects. So each row under each time step of Table 2 forms a transaction. We looked at 8x8x8 neighborhood box and 16x16x16 neighborhood box, and explored several ways to construct attributes from the grid points in the box. We had 1,000 time steps of the simulation data. We sampled every 100 time steps to have total 30,720 transactions (48*64 transactions per time step). For these transactions, we took the first 10,000 transactions as the training set and the last 10,000 transactions as the test set.

This data set has a large variation in the size of each class. In particular, about 89.1% of the transactions have class 0 (no velocity movement), 4.2% have class -1 (downward velocity movement), and 6.7% have class 1 (upward velocity movement). With this class distribution, any simple classification algorithm that classifies everything to be class 0 has a classification accuracy close to 90%. Hence, classification accuracy is not a good measure to evaluate classifier. In such data sets, precision and recall are considered more useful evaluation measures [Sal89]. Recall and precision for a class are defined as:

$$recall = \frac{\text{number of records correctly predicted to be of the class}}{\text{number of records in the class}}$$

$$precision = \frac{\text{number of records correctly predicted to be of the class}}{\text{number of records predicted to be of the class}}$$

Table 3 shows a confusion matrix for a three class problem. Rows A, B, and C show the number of records that belong to class A, B, and C, respectively. Columns A, B, and C shows the number of records classified as belong to classes A, B, and C, respectively. Recall and precision values of class A are $i/(i+j+k)$ and $i/(i+l+o)$, respectively. Note that it is possible to obtain 100% recall for any given class A by using a trivial model that classifies all records to be of class A. But this makes the precision of class A quite poor, particularly if class A is smaller than the other classes. Similarly, it is possible to obtain high values of precision for class A by choosing a highly conservative model for class A. But this makes the recall of class A quite poor.

The precision-recall break even point is the value at which precision and recall are equal [YP97], and is often used as a performance metric, especially for comparing classification algorithms that allow the tradeoff of precision and recall values.

In our experiments, we used C4.5rules [Qui93] and SVM (support-vector machines) [CV95, Joa98] for building explanatory and predictive

Table 3. A sample confusion matrix.

Original class	Predicted class A	B	C
A	i	j	k
B	l	m	n
C	o	p	q

models. We chose C4.5rules because it provides explicit rules that scientists can examine and validate. We chose SVM for its high quality classification results in data sets with many attributes [Joa98].

4.1. C4.5rules Results

8x8x8 neighborhood box. The first 5 data sets were generated using 8x8x8 neighborhood box. The first data set, DS:8x8x8:1x1x1, has 512 binary variables where each variable denotes the presence or absence of the swirl in the corresponding grid point. The second data set, DS:8x8x8:2x2x2, has 64 variables where each variable corresponds to the number of grids with swirl value in 2x2x2 block. The third data set, DS:8x8x8:4x4x4, has 8 variables where each variable corresponds to the number of grids with swirl value in 4x4x4 block. The fourth data set, DS:8x8x8:8x8x8 has 1 variable that corresponds to the number of grids with swirl value in the whole block.

The confusion matrices and precision and recall values of each class for 8x8x8:1x1x1 are shown in Table 4 for both the training set and test set. Most of the entries in the Table 4 for both training and test sets are on the diagonal. Classification accuracy on the training and test sets are 96.1% and 90.2% respectively. The classification accuracy is high because the largest of the three classes (class 0) has high recall and high precision for both training and test sets. For both class -1 and 1, precision values are quite high (98.0% for each class) and recall values are quite respectable (70.0% and 64.0%, respectively) for the training set. But on the test set, these values drop significantly. Nevertheless, these values are considerably better than those expected from a random model.

The precision and recall of DS:8x8x8:1x1x1 on test set was the best among the data sets generated within 8x8x8 neighboring box. The observed trend is that the performance decreases as the number of variables decreases.

16x16x16 neighborhood box. Next, we expanded the neighborhood box to study the impact of neighborhood size. Data sets were gen-

Table 4. Confusion matrix and precision and recall of each class for datasets generated from 8x8x8 neighborhood box.

DS:8x8x8:1x1x1 with 512 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
296	128	1	(a): class -1	0.98	0.70	97	283	15	(a): class -1	0.44	0.25
6	8895	7	(b): class 0	0.96	1.00	90	8717	129	(b): class 0	0.92	0.98
1	242	424	(c): class 1	0.98	0.64	32	430	207	(c): class 1	0.59	0.31

DS:8x8x8:2x2x2 with 64 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
339	86	0	(a): class -1	0.95	0.80	98	281	16	(a): class -1	0.35	0.25
15	8874	19	(b): class 0	0.97	1.00	165	8552	219	(b): class 0	0.93	0.96
1	159	507	(c): class 1	0.96	0.76	18	398	253	(c): class 1	0.52	0.38

DS:8x8x8:4x4x4 with 8 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
335	90	0	(a): class -1	0.95	0.79	81	292	22	(a): class -1	0.28	0.21
17	8862	29	(b): class 0	0.98,	0.99	173	8408	355	(b): class 0	0.92	0.94
1	130	536	(c): class 1	0.95,	0.80	37	476	156	(c): class 1	0.29	0.23

DS:8x8x8:8x8x8 with 1 variable

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
13	399	13	(a): class -1	0.33	0.03	0	395	0	(a): class -1	1.00	0.00
20	8879	9	(b): class 0	0.90	1.00	0	8929	7	(b): class 0	0.89	1.00
7	639	21	(c): class 1	0.49	0.03	0	667	2	(c): class 1	0.22	0.00

erated using a 16x16x16 neighborhood box. Similar to 8x8x8 neighborhood box we generated data sets DS:16x16x16:1x1x1 with 4096 variables, DS:16x16x16:2x2x2 with 512 variables, DS:16x16x16:4x4x4 with 64 variables, DS:16x16x16:8x8x8 with 8 variables, and DS:16x16x16:16x16x16 with 1 variable. The confusion matrices and precision and recall values of each class are shown in Table 5. Within these data sets, we observe overall trends that are similar to data sets with 8x8x8 neighboring region. The precision and recall on the test set shows that DS:16x16x16:1x1x1 provides better result, compared to the data sets with 8x8x8 neighboring regions. For example, on the test set, the precision for DS:16x16x16:1x1x1 is 0.63 for both class -1 and 1, whereas for DS:8x8x8:1x1x1, these values are 0.44 and 0.59, respectively. Similarly, the recall on DS:16x16x16:1x1x1 is 0.40 and 0.42 for class -1 and 1, respectively, whereas for DS:8x8x8:1x1x1, those values are 0.25 and 0.31, respectively.

16x16x16 neighborhood shell. The next data sets were generated by considering neighboring shells surrounding a given grid point. The confusion matrices and precision and recall values of each class are shown Table 6.

Table 5. Confusion matrix and precision and recall of each class for datasets generated from 16x16x16 neighborhood box.

DS:16x16x16:1x1x1 with 4096 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
330	95	0	(a): class -1	0.99	0.78	159	231	5	(a): class -1	0.63	0.40
4	8888	16	(b): class 0	0.98	1.00	85	8693	158	(b): class 0	0.93	0.97
1	115	551	(c): class 1	0.97	0.83	10	376	283	(c): class 1	0.63	0.42

DS:16x16x16:2x2x2 with 512 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
392	32	1	(a): class -1	0.98	0.92	120	257	18	(a): class -1	0.53	0.30
6	8890	12	(b): class 0	0.98	1.00	97	8704	135	(b): class 0	0.92	0.97
1	119	547	(c): class 1	0.98	0.82	8	463	198	(c): class 1	0.56	0.30

DS:16x16x16:4x4x4 with 64 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
355	70	(a):	(a): class -1	0.95	0.84	88	296	11	(a): class -1	0.43	0.22
16	8874	18	(b): class 0	0.98	1.00	103	8618	215	(b): class 0	0.91	0.96
1	110	556	(c): class 1	0.97	0.83	14	512	143	(c): class 1	0.39	0.21

DS:16x16x16:8x8x8 with 8 variables

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
359	65	1	(a): class -1	0.87	0.84	51	318	26	(a): class -1	0.18	0.13
53	8826	29	(b): class 0	0.97	0.99	211	8279	446	(b): class 0	0.91	0.93
1	166	500	(c): class 1	0.94	0.75	25	547	97	(c): class 1	0.17	0.14

DS:16x16x16:16x16x16 with one variable

training set						test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
12	408	5	(a): class -1	0.67	0.03	0	395	0	(a): class -1	0.00	0.00
6	8888	14	(b): class 0	0.89	1.00	2	8934	0	(b): class 0	0.89	1.00
0	644	23	(c): class 1	0.55	0.03	0	669	0	(c): class 1	1.00	0.00

Table 6. Confusion matrix and precision and recall of each class for datasets generated from 16x16x16 neighborhood box.

DS:16x16x16:radius-1 with 6 variables

training set						test set					
(a)	(b)	(c)	<-predicted class	precision	recall	(a)	(b)	(c)	<-predicted class	precision	recall
326	97	2	(a): class -1	0.95	0.77	64	280	51	(a): class -1	0.18	0.16
15	8864	29	(b): class 0	0.97	1.00	255	8257	424	(b): class 0	0.92	0.92
1	158	508	(c): class 1	0.94	0.76	32	466	171	(c): class 1	0.26	0.26

DS:16x16x16:radius-2 with 12 variables

training set						test set					
(a)	(b)	(c)	<-predicted class	precision	recall	(a)	(b)	(c)	<-predicted class	precision	recall
366	58	1	(a): class -1	0.94	0.86	84	263	48	(a): class -1	0.27	0.21
24	8864	20	(b): class 0	0.98	1.00	207	8394	335	(b): class 0	0.92	0.94
1	134	532	(c): class 1	0.96	0.80	17	439	213	(c): class 1	0.36	0.32

DS:16x16x16:radius-4 with 24 variables

training set						test set					
(a)	(b)	(c)	<-predicted class	precision	recall	(a)	(b)	(c)	<-predicted class	precision	recall
360	65	0	(a): class -1	0.96	0.85	93	284	18	(a): class -1	0.35	0.24
12	8878	18	(b): class 0	0.98	1.00	156	8511	269	(b): class 0	0.92	0.95
2	141	524	(c): class 1	0.97	0.79	20	478	171	(c): class 1	0.37	0.26

DS:16x16x16:radius-8 with 48 variables

training set						test set					
(a)	(b)	(c)	<-predicted class	precision	recall	(a)	(b)	(c)	<-predicted class	precision	recall
382	40	3	(a): class -1	0.97	0.90	127	259	9	(a): class -1	0.42	0.32
12	8758	138	(b): class 0	0.98	0.98	156	8532	248	(b): class 0	0.93	0.95
1	180	486	(c): class 1	0.78	0.73	17	415	237	(c): class 1	0.48	0.35

We can observe similar trend on these data sets. For instance, the precision and recall on test sets improve as more number of variables are used, i.e., DS:16x16x16:radius-1 has the worst performance and DS:16x16 x16:radius-8 has the best performance. Compared to the results of data set DS:16x16x16:1x1x1, the precision and recall for DS:16x16x16:radius-8 on test set are worse. However, these two data sets are not directly comparable because there is a large difference in the number of variables. A fairer comparison is between DS:16x16x16:4x4x4 (that has 64 variables) and DS:16x16x16:radius-8 (that has 48 variables). Note that DS:16x16x16:radius-8 has much better overall performance than DS:16x16x16:4x4x4 in terms of precision and recall for class -1 and 1. This suggests that modeling based on shells instead of boxes might be a more robust approach.

Top Five Rules of Each Class. Table 7 shows the top five rules for each class with respect to the training set support count and the accuracy of each rule in DS:16x16x16:1x1x1. The support count of a rule is defined as the number of samples that are classified using the rule. This table also shows the support count and accuracy of a rule on

Table 7. Statistics of top 5 rules of each class in DS:16x16x16:1x1x1.

Rule		Training Set		Test Set		Test Set Distribution		
Number	Class	Support	Accuracy	Support	Accuracy	-1	0	1
91	-1	21	1.00	14	0.79	11	3	0
147	-1	19	1.00	12	1.00	12	0	0
82	-1	18	0.94	11	0.45	5	6	0
63	-1	17	1.00	16	0.44	7	9	0
279	-1	17	1.00	11	0.27	3	8	0
142	0	6842	0.98	7358	0.96	140	7058	160
168	0	928	0.97	793	0.91	26	718	49
215	0	574	0.96	535	0.81	28	433	74
233	0	317	0.99	344	0.78	14	269	61
223	0	109	0.98	56	1.00	0	56	0
247	1	33	1.00	32	0.59	0	13	19
263	1	28	0.96	26	0.96	0	1	25
271	1	23	1.00	20	0.55	3	6	11
250	1	22	1.00	24	0.88	0	3	21
171	1	21	1.00	17	0.82	0	3	14

the test set and the class distribution of test set that are classified using the rule.

This table shows that class 0 rules are quite good - they have high support and confidence on both training and test sets. Just the top three rules are able to cover over 90% of the records in class 0. This shows that it is possible to build a compact prediction model for class 0. Rules for class 1 and -1 covers much smaller fraction of their respective classes, and the accuracy of these rules tends to be significantly lower on the test data set than on the training data set. This shows that difficulty inherent in building models for class 1 and -1. Nevertheless, most of the inaccuracy of these rules is due to misclassification to class 0 and not due to misclassification to the opposite class.

Figure 4 shows some of the rules of Table 7 and some of the rules from DS:16x16x16:radius-2 data set. $sc(x, y, z)$ refers to the swirl count at location (x, y, z) with respect to the grid point. $radius - below(a : b)$ refers to the swirl count in shells below the XY plane with radius between a and b. Similarly, $radius - above(c : d)$ refers to the swirl count in shells above XY plane with radius between c and d. The two ratios shown at the end of each rule show the coverage and confidence of this rule in training set and the test set respectively. C/T shows that this particular rule covers T total transactions (i.e., T transactions satisfies the precedent of this rule) and C transactions are classified correctly.

Rules in this figure show that the upward and downward velocity movement has a high correlation with the swirl activities of its neighbors. For example, rule 142 of Figure 4 shows that when there is no

```
Rule 263:
  sc(0, -1, 0) > 0
  sc(0, 2, 5) > 0
  sc(-4, -3, -8) <= 0
  sc(6, 0, -5) <= 0
  sc(-2, 0, -8) <= 0
  ->  class 1  (27/28, 25/26)

Rule 91:
  sc(-1, 3, 1) > 0
  sc(0, 4, 5) > 0
  sc(-2, -4, 1) > 0
  ->  class -1  (21/21, 11/14)

Rule 142:
  sc(0, -1, 0) <= 0
  sc(0, 0, 2) <= 0
  sc(0, 1, 2) <= 0
  sc(-4, 5, 4) <= 0
  sc(5, 4, 5) <= 0
  sc(5, 5, 6) <= 0
  sc(0, 6, 7) <= 0
  sc(1, 0, 2) <= 0
  sc(7, 1, 7) <= 0
  ->  class 0
     (6727/6842, 7058/7358)
```

```
Rule 382:
  radius-below(0:2) > 4
  radius-below(6:8) <= 37
  radius-below(14:16) <= 1
  radius-above(4:6) > 15
  radius-above(6:8) <= 40
  radius-above(10:12) > 1
  ->  class 1  (29/29, 20/26)

Rule 367:
  radius-below(2:4) > 26
  radius-below(10:12) > 33
  radius-above(0:2) <= 6
  radius-above(2:4) > 11
  radius-above(6:8) > 44
  radius-above(6:8) <= 68
  radius-above(8:10) > 30
  radius-above(10:12) > 16
  ->  class -1  (42/43, 10/10)

Rule 263:
  radius-below(0:1) <= 2
  radius-above(2:3) <= 1
  ->  class 0  (1339/1346, 1670/1732)
```

Figure 4. Sample C4.5 rules from datasets DS:16x16x16:1x1x1 and DS:16x16x16:radius-2.

swirl activity around the gird, there is no velocity movement (class 0). Similarly, rule 263 shows that when swirl count is small in shell below (radius 0 and 1) and in shell above (radius 2 and 3), there is no velocity movement (class 0). Rule 263 shows that upward velocity exists (class 1) when there are swirl activities in some grid points and no swirl activities in some other grid points. Note that some of the conditions of the rule are due to C4.5rules generation side effects. We removed the last three conditions of the rule and used the reduced rule on the test set. The result shows that only three more class 0 test cases are classified as class 1 with this rule. In other words, rule 263 simply shows that upward velocity exists when there are swirl activities in some grid points. We did a similar experiment with Rule 367. We removed conditions that has an expression swirl count is less than certain constant. The result shows that only one class 0 test case is classified as class 1 with this new rule. Once again, the reduced rule says that when there are many swirl counts in neighboring shells, there is downward velocity movement (class -1).

Table 8. Confusion matrix of dataset DS:16x16x16:1x1x1 and DS:16x16x16:radius-2 on test set close to the training set and the original test set.

DS:16x16x16:1x1x1

test set close by						original test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
216	191	8	(a): class -1	0.69	0.52	159	231	5	(a): class -1	0.63	0.40
87	8660	184	(b): class 0	0.95	0.97	85	8693	158	(b): class 0	0.93	0.97
8	289	357	(c): class 1	0.65	0.55	10	376	283	(c): class 1	0.63	0.42

DS:16x16x16:radius-2

test set close by						original test set					
(a)	(b)	(c)	<-predicated class	precision	recall	(a)	(b)	(c)	<-predicated class	precision	recall
139	248	39	(a): class -1	0.39	0.33	84	263	48	(a): class -1	0.27	0.21
211	9087	319	(b): class 0	0.93	0.94	207	8394	335	(b): class 0	0.92	0.94
10	393	274	(c): class 1	0.43	0.40	17	439	213	(c): class 1	0.36	0.32

Effect of temporal autocorrelation. In the experiments shown above, we have chosen to train on the data set in the first one third of the simulation time frame and test on the data set in the last one third of the simulation time frame. If we had randomly chosen half of data from the same time frame as a training set and tested on the remaining half, the precision/recall on test set would have been very close to that of the training set. This is due to the temporal autocorrelation present in the simulation data sets, i.e., burst events tend to occur together in a small time span. As the test set is moved further apart from the training set in terms of time steps, the effect of autocorrelation diminishes. Table 8 shows a comparison of the confusion matrix from a test set that immediately follows the training set in time and the original test set of dataset DS:16x16x16:1x1x1 and DS:16x16x16:radius-2. This table shows that the classification accuracy, precision and recall on test set close to the training set are much better than the original test set that are quite far away from the training set in time.

4.2. SVM Results

We ran SVM-Light [Joa98], which is a publicly available SVM binary classifier, on the data sets used for C4.5rules. Table 9 shows precision, recall, and precision-recall break even point for these data sets. The precision and recall is computed based on the results of using default parameter settings, and the break even point was computed by changing classification threshold on the same result.

Compared to the C4.5rules results, SVM results have much better precision and recall. For example on 16x16x16:1x1x1 data set, the precision and recall of SVM on class 1 was 0.94 and 0.52 respectively and precision-recall break even point was 0.75. C4.5rules had precision and recall of 0.63 and 0.42 respectively. The precision and recall on class

Table 9. Precision and recall results of SVM on various data sets.

	class 1			class -1		
dataset	precision	recall	breakeven point	precions	recall	breakeven point
8x8x8:1x1x1	0.79	0.27	0.56	0.97	0.15	0.41
16x16x16:1x1x1	0.94	0.52	0.75	0.96	0.46	0.73
32x32x32:1x1x1	0.93	0.62	0.81	0.98	0.64	0.81

-1 was 0.96 and 0.46 and precision/recall break-even point was 0.73. C4.5rules had precision and recall of 0.63 and 0.40. This shows that SVM has a significant performance advantage over C4.5rules.

In the C4.5rules, we saw that the performance improved from DS:8x8x8 :1x1x1 to DS:16x16x16:1x1x1. We wanted to try C4.5rules on the DS:32x 32x32:1x1x1 data set, but we ran out of memory because C4.5rules does not support a sparse data format. Since SVM-Light can handle a sparse data format, we tried SVM on DS:32x32x32:1x1x1 data set. On this data set, the precision and recall on class 1 was 0.93 and 0.62 and precision-recall break even point was 0.81. The precision and recall on class -1 was 0.98 and 0.64 and precision-recall break even point was 0.81. This shows that the classification quality improves as we consider larger spatial neighbors. Furthermore, the precision-recall break even point of 0.81 for both classes are very good considering the limited information we have in the model.

5. Concluding Remarks and Future Work

In this chapter, we have proposed several data modeling methods to incorporate the spatial and temporal features of turbulent flow simulation data. We have also applied classification rules algorithm C4.5rules and SVM on the turbulent flow simulation data to identify upward or downward velocity movements of the flow close to the wall. Some of the rules discovered by C4.5rules have good predictive power in test set and can potentially provide an explanatory model for the burst event in the turbulent flow simulation. The SVM results also show that the bursting events can be predicted with considerable accuracy. One of disadvantage of SVM in general is that it is difficult to identify important features used in the classification model. Since the SVM results are significantly better than C4.5rules results, it will be worth investigating some ways to extract important features or rules from the support vector of the SVM.

The experiments reported in this chapter is a small step toward the application of data mining techniques in the scientific simulation data. Several outstanding research issues still remain. Note that the precision and recall for class -1 and 1 are significantly lower on the test data than in the training data, except when the number of independent variables used is quite large. Since there is a temporal autocorrelation in the data, there is a possibility that much of the predictive accuracy is due to the fact that the frames used for generating the test data are not sufficiently different than the one used for generating the training data. (We note that in our visual inspection, we found the training and test frames to be quite different.) Another possible explanation is that the current modeling of data sets is not not sufficient to capture the underlying mechanism of bursting event. We might need to look at different box sizes for collecting variables, need different variables other than swirl strength, need to have higher level features, or need temporal features in the transactions. Finally, we may not have a sufficient variety of training samples to cover different patterns that may arise in turbulent flow simulation, i.e., we might need a greater variety of simulation data sets to capture the general model.

Acknowledgments

The authors wish to thank Drs. I. Marusic, P. Martin, and G. Candler for providing data sets and domain knowledge for our project.

This work was supported by NSF ACI-9982274, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE grant LLNL/DOE B347714, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute.

References

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Monterrey, CA, 1984.

[Coh95] W.W. Cohen. Fast effective rule induction. In *Proc. of the Twelfth International Conference on Machine Learning*, 1995.

[CV95] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[DK01] Mukund Deshpande and George Karypis. Selective markov models for predicting web-page accesses. In *First Siam International Conference on Data Mining*, 2001.

[DMT94] D.J. Spiegelhalter D. Michie and C.C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[HKK01] E.H. Han, G. Karypis, and V. Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In David Wai-Lok Cheung, Graham J. Williams, and Qing Li, editors, *PAKDD*, volume 2035 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

[Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.

[MCI+01] I. Marusic, G.V. Candler, V. Interrante, P.K. Subbareddy, and A. Moss. Real time feature extraction for the analysis of turbulent flows. In R. Grossman, C. Kamath, Ph. Kegelmeyer, V. Kumar, and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications.* Kluwer, 2001.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[Sal89] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[YP97] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.

Chapter 15

EVITA - EFFICIENT VISUALIZATION AND INTERROGATION OF TERA-SCALE DATA

Raghu Machiraju, James E. Fowler, David Thompson, Bharat Soni, and Will Schroeder

Abstract Large-scale computational simulations of physical phenomena produce data of unprecedented size (terabyte and petabyte range). Unfortunately, development of appropriate data management and visualization techniques has not kept pace with the growth in size and complexity of such datasets. To address these issues, we are developing a prototype, integrated system (EVITA) to facilitate exploration of tera-scale datasets. The cornerstone of the EVITA system is a representational scheme that allows ranked access to macroscopic features in the dataset. The data and grid are transformed using wavelet techniques while a feature-detection algorithm is used to identify and rank contextually significant features directly in the wavelet domain. The most significant parts of the dataset are thus available for detailed examination in a progressive fashion. The work presented here is similar in essence to much of the work in the traditional data-mining community. We first describe the basic system and follow with a discussion of ongoing work, focusing on efforts in multiscale feature detection and progressive access. Finally, we demonstrate the system for a two-dimensional vector field derived from an oceanographic dataset.

Keywords: Feature detection, vector fields, compression, embedded and progressive access, large datasets, tera-scale visualization

1. Introduction

Computational field simulation (CFS) [Jam92, DTZ$^+$97, KP97, SG95] is the process of simulating complex physical phenomena via computation for analysis, design, or investigation of fundamental physical phenomena. The computed results are then interrogated for analysis and validation through visualization. In such applications, the objective of visualization is to interactively transform data into pictorial form in a manner which fosters discovery by the viewer.

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 257–279.
© 2001 *Kluwer Academic Publishers.*

CFS routinely produces data of unprecedented size (terabyte and petabyte range) which introduces new challenges to data management and visualization. Coupled with growth in data size is the need for increased realism due to more demanding user expectations. Unfortunately, it is often difficult to perform even simple visualization tasks on extremely large datasets since the development of appropriate data-management techniques has not kept pace with the growth in size and complexity of datasets [SvR98]. For many simulations, the storage of the data itself, not to mention transmission over a limited-bandwidth network, is a serious problem.

The research reported here attempts to respond to the challenges posed by extremely large datasets. We believe the key technology needed to facilitate tera-scale visualization is *ranked access of relevant features in compressed representations*. In tera-scale visualization, locating important features of the dataset first is paramount for effective data exploration. For practical reasons arising primarily from resource limitations (e.g., CPU speed, network bandwidth, main-memory availability), only parts of a dataset may be accessed at any given time (i.e., out-of-core analysis and visualization). Thus, it is necessary to create schemes that allow rapid access to significant information by considering only a small portion of the dataset. To this end, automatic detection and ranking of contextually significant features in datasets is of critical importance. Furthermore, often both focus and context views of datasets [Fur86, RCM93, JS98] are required; i.e., certain elements of interest (focus) are enhanced while the remainder of the domain provides a non-distracting context. Under this paradigm, the researcher attempts to search for useful information in the unfamiliar territory of the dataset in question by exploiting a fish-eye focused view. Additionally, based on discoveries along the way, the objective of the search may change, since better understanding of the dataset allows for more refined searches.

The data-exploration and visualization scenario described above is not markedly different from what occurs during a typical data-mining session. Determining correlation and inherent relationships between various entities is the primary goal of a data-mining exercise. These in turn produce a better understanding of the processes generating the data. Similarly, a meaningful scientific-visualization exercise should facilitate discovery of important phenomena. However, essential differences do exist between the scientific domain and typical data-mining applications. CFS data is not abstract and, in fact, is deterministic. The generation of the dataset is dictated by mathematical models representing physical processes. As a result, detection of essential spatial and temporal relationships (and hence coherent structures) can be conducted with deterministic techniques as opposed to statistical and sampling techniques characteristic of traditional data mining. It should be noted that there do exist cases in which non-deterministic and statistical techniques can provide an understanding of certain physical phenomena, e.g., turbulence. However, most of these techniques are essentially deterministic in nature since realistic physical models are available. Additionally, an important reason for the data explosion in typical data-mining applications is the large parameter space in which the entities exist while, in CFS applications, the disparate spatial and temporal scales required to simulate complex physical phenomenon dictate the growth of the datasets.

For effective tera-scale visualization, data compression is unavoidable given the enormous quantities of data involved. For generality, compressed representations should be appropriate for both scalar and vector fields, with time-varying fields being of particular importance as the time-varying nature of such datasets often yields extremely large sizes. Additional constraints on compression algorithms are as follows. First, it is often desired, for reasons of efficiency, that derived quantities (e.g., gradients, vorticity), needed by various visualization and analysis algorithms, be computed in the compressed domain; therefore, compressed representations should be amenable to such operations. Secondly, perfect reconstruction should be permitted. That is, the compression operation should result in no net loss of information and, if desired, the original dataset should be recoverable from the compressed representation with no loss of data. Finally, and most importantly for ranked-access systems, visualization should be facilitated by an incomplete reconstruction of the dataset. That is, the compressed representation must support browsing by partial reconstruction of the original dataset as defined by viewer requirements. Such browsing capability can be implemented with a client-server model—the client delineates certain *regions-of-interest* (ROIs), while the server manipulates the flow of transmitted data so as to send information from selected ROIs first. In this way, the browsing tool is amenable to distributed- or even remote-visualization uses.

An outline of the remainder of our discussion is as follows. In Section 2, we overview our prototype system for visualizing large datasets, EVITA, and we provide details on a method to create a significance map and obtain rankings for the various ROIs in the dataset in Section 3. Compression and coding of vector fields and their associated grids are discussed in Section 4, while Section 5 outlines a vision for a user interface. Finally, in Section 6, we present preliminary results for our system for a two-dimensional vector-field dataset. We note that are discussion here is of a work-in-progress and much on-going effort is currently being devoted to expand on the reported results.

2. System Description

We are currently developing the EVITA system to facilitate exploration of tera-scale datasets. A scalable system amenable to distributed implementation, EVITA is concerned with the storage and representation of very large datasets, their analysis and manipulation, and rendering and presentation of the resulting images. Viable tera-scale visualization is possible only through a confluence of knowledge from visualization, coding theory, geometry, and application-specific domains. Consequently, the EVITA system is a sophisticated amalgamation of diverse techniques and will provide an efficient interrogative tool for tera-scale visualization.

In designing the EVITA system, we have focused on time-varying, computational-fluid dynamic/hydrodynamic, environmental-science, and oceanographic datasets on structured rectilinear or curvilinear grids. Many computational investigations are conducted on such structured sampling regimes. Although these structured grids are less complex than unstructured meshes, there are still many difficult and unresolved problems relating to the compression and transmission of data sampled on structured grids. Additionally, wavelet theory has not been fully developed for unstructured grids. Some promising work by Guskov *et al.* exists for surface meshes [GSS99]; however, significant further development is still required before such techniques can be deployed for compression purposes. To reiterate, our focus is on ranked access of relevant features in compressed representations.

Figure 1 shows a block diagram of the EVITA system which is comprised of three main components: an offline *preprocessor*, a *server*, and a *client*. The preprocessor takes the original dataset and its associated grid to produce a compact representation. Although this compression process is expected to be quite time consuming due to a computationally expensive feature-detection step, this complex computation is performed only once, offline. The compressed bitstream $\tilde{B}$ resulting from this offline preprocessing is produced under a fixed priority schedule permitting a suitable visualization of the dataset. It is important that features to be detected be chosen with utmost care such that they are sufficiently general for a variety of visualizations.

The interactive ROI operation required of the system is produced by the server, a thread running during the visualization session. The goal of the server is to relay the compressed dataset to the client, taking into account ROIs as specified by the user. While the visualization process is underway, the server accepts ROI information in real time from the client and dynamically effectuates the necessary reorganization of bitstream $\tilde{B}$ to produce the desired priority ranking. In essence, the ROI priority ranking is accomplished by a mere reordering of the bits of the bitstream; as such, this operation is computationally simple and possible at interactive rates (as opposed to the encoding operation itself which is confined to the offline domain).

The final component of the system, the client, decodes the reorganized bitstream arriving from the server, produces the visualization, and accepts user feedback in the form of ROI specification. The decoding process produces frequent reconstructions of the dataset as additional information is received

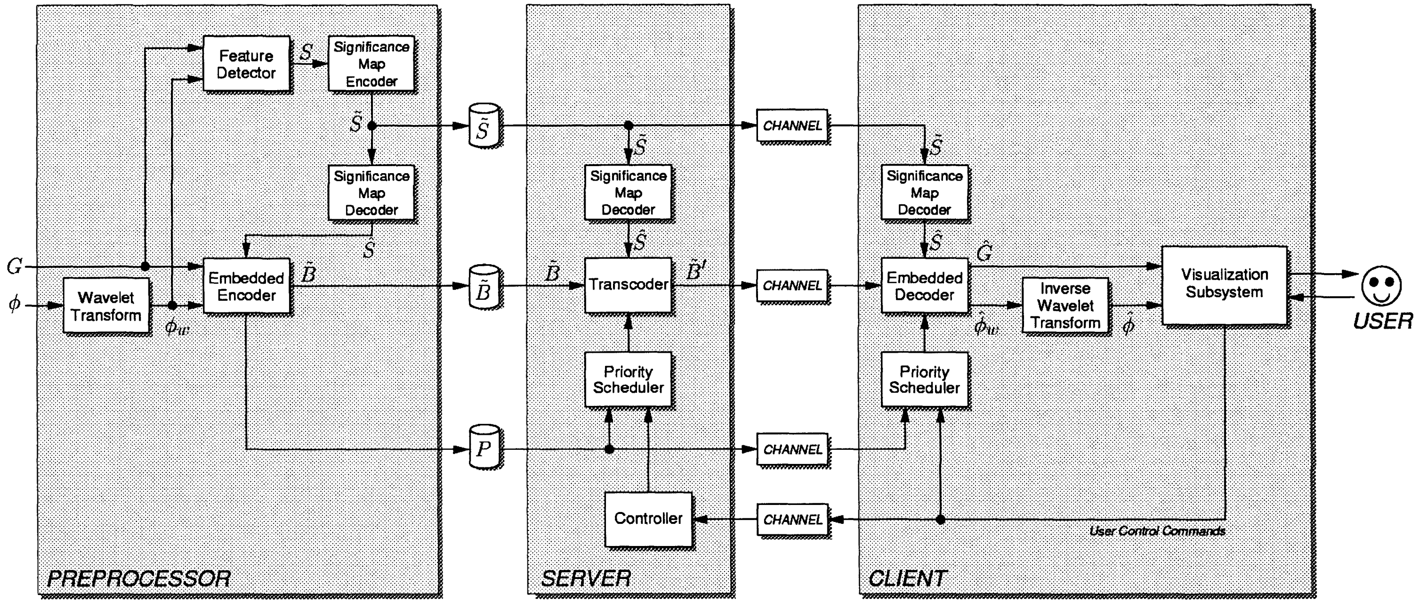

G
ϕ
Wavelet Transform
ϕ_w
Feature Detector
S
Significance Map Encoder
$\tilde{S}$
Significance Map Decoder
S
Embedded Encoder
$\tilde{B}$
PREPROCESSOR
$\tilde{S}$
$\tilde{B}$
P
$\tilde{S}$
Significance Map Decoder
$\hat{S}$
$\tilde{B}$
Transcoder
$\tilde{B}'$
Priority Scheduler
Controller
SERVER
CHANNEL
CHANNEL
CHANNEL
CHANNEL
$\tilde{S}$
Significance Map Decoder
$\hat{S}$
$\hat{G}$
Embedded Decoder
$\hat{\phi}_w$
Inverse Wavelet Transform
$\hat{\phi}$
Visualization Subsystem
Priority Scheduler
User Control Commands
CLIENT
USER

Figure 1. EVITA - A Prototype System for Effective Visualization and Interrogation of Tera-scale Datasets. Legend: ϕ scalar or vector data field, G discrete grid, S significance map, $\tilde{B}$ bitstream. $\tilde{x}$ denotes encoded (compressed) quantities, $\hat{x}$ reconstructed (decompressed) quantities, x' reordered (transcoded) quantities, and x_w wavelet-domain quantities.

from the server; these reconstructions are successively fed to the visualization subsystem which updates the current rendering of the dataset.

Development of the system is based upon advances in wavelet transforms, scalar- and vector-field analysis, progressive and embedded encoding, and visualization toolkits. The EVITA system can provide a general visualization environment for any tera-scale dataset with underlying structure similar to that for which the system is designed. The application-specific components of the system lie merely in the feature-detection algorithm and in the generation of the significance map—aside from these two modules, the system, as proposed, is amenable to many visualization tasks.

In what follows, we present an overview of the EVITA system, describing ongoing design efforts in significance-map generation, data-grid coding, and visualization interfaces, and presenting preliminary results obtained from the current system implementation.

3. Significance Map Generation

Mechanisms for location of application-specific features in the data and generation of a ranking of these identified features is critical to the success of the EVITA system. We address the issue of ranked access with a procedure that culminates in the generation of a four-dimensional significance map. The first step is identification of the desired features within the dataset, which then yields a segmentation of the dataset into ROIs. Consideration of time-varying data complicates the feature-identification process since, once identified, features must be tracked across multiple time steps. After segmentation, ROIs are then ranked according to appropriate criteria. In the completed system, a four-dimensional significance map will be generated that will establish the baseline priority schedule for dataset encoding. For simplicity, we describe here techniques to generate a two-dimensional significance map for a steady-state, two-dimensional vector field. The data employed here is from the equatorial region of the Pacific Ocean generated using the Naval Layered Ocean Model [Wal99]. To illustrate the operation of the EVITA system, the features we consider here are regions of swirling flow. However, our approach is applicable to ROIs corresponding to any general feature identifiable from a scalar field.

Linear-Lifting Scheme for Vectors. An extension of the linear-lifting scheme to perform a wavelet transform for vector data is discussed in this section. Intentionally, a simple scheme was chosen for this effort. In general, the lifting scheme consists of three steps: split, predict, and update (see [Swe97] for further detail). For linear lifting, the filter coefficients employed during the predict stage are $g = \{0.5, 1.0, 0.5\}$, while the coefficients for the update stage are $h = \{0.25, 0.25\}$. We use a critically sampled (i.e., decimated) transform and achieve a multi-dimensional transform by applying a sequence of one-dimensional wavelet transforms along each dimension in succession. For example, for a two-dimensional dataset, one-dimensional transforms are applied first along rows and then along columns, and each transform level yields four subbands corresponding to a smooth decimated representation and details along horizontal, vertical, and diagonal orientations. Transforms for higher-dimensional data proceed similarly. For vector-valued data, the wavelet transform is applied to each component of the vector in an independent fashion.

Feature Detection in Flow Datasets. Computational fluid dynamics (CFD) datasets typically contain one or more features of interest. These features, with specific fluid-dynamic characteristics, include shocks, vortices, flow separations, etc. The focus here is on features characterized by a swirling motion suggested by rotation of fluid particles about a common center, either fixed or moving.

Berdahl and Thompson define a derived scalar quantity called *swirl* which can be used to identify features characterized by swirling flow such as vortices [BT93]. In this procedure, a scalar value based on the local velocity and velocity gradients is assigned to each field point. The swirl value is interpreted as the tendency for the fluid to swirl at a given point. Contiguous regions of nonzero swirl values can therefore be thought of as distinct features. Core regions of vortices are characterized by larger swirl values. This methodology can be contrasted with those methods that determine critical points and then connect them with curves [HH89] or seek to locate vortex core regions [SH95].

A three-dimensional velocity field is characterized by the vector $\mathbf{V} = [u, v, w]^T$ at each point (x, y, z) in the domain. The velocity gradient tensor describes the spatial variation of the velocity field at a point and is given by:

$$\Gamma = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} . \tag{15.1}$$

A necessary condition for the occurrence of swirling motion is the existence of a region of complex conjugate pairs of the eigenvalues of Γ. Associated with the complex eigenvalues is a cyclical motion with a period that is proportional to the magnitude of the imaginary component of the eigenvalues. This period can be computed using

$$t_{orbit} = \frac{2\pi}{|Im\,(\lambda_{1,2})\,|} \tag{15.2}$$

where $Im\left(\lambda_{1,2}\right)$ is the imaginary part of the complex conjugate pair of eigenvalues.

An additional consideration is that a fluid particle may convect out of a region of complex eigenvalues before swirling motion is apparent. Computing the time required to traverse the region of complex eigenvalues is problematic since it would require a detailed particle-trajectory computation. If d is the characteristic length associated with the size of the region of complex eigenvalues and V_{conv} is the magnitude of the average velocity along d, then a simple estimate for the convection time is given by

$$t_{conv} = \frac{d}{V_{conv}} \ . \tag{15.3}$$

The convection velocity V_{conv} is taken to be the velocity in the plane containing the swirling motion, i.e., the plane normal to the real eigenvectors at the point in question. The length d is typically taken to be unity.

The swirl parameter τ is then defined as the ratio of the time for a fluid particle to convect through the region of complex eigenvalues to the orbit time

$$\tau = \frac{t_{conv}}{t_{orbit}} = \frac{\left|Im\left(\lambda_{1,2}\right)\right|}{2\pi V_{conv}} \ . \tag{15.4}$$

For small values of τ, the fluid convects too rapidly through the region of complex eigenvalues to be captured in the swirling motion. In regions of large τ, the fluid is trapped in a swirling motion. It is to be noted that the swirl values in this work are calculated as the logarithm of the formula given in Equation 15.4. More details can be found in [BT93]. Figures 2(a) and (b) depict the magnitude of the original vector field and the corresponding swirl field, respectively, for a portion of the equatorial region of the Pacific Ocean [Wal99]. Of particular significance is the fact that regions of high velocity, as denoted by lighter shades in Figure 2(a), do not necessarily correlate with regions where the fluid is swirling.

Multi-Scale Significance Map Generation. The image in Figure 2(b), which shows the swirl parameter calculated on a two-dimensional grid of dimension $N \times N$, is called a significance map and is denoted by

$$S_{single} = \left\{ s\left(i,j\right) | i,j = 0,\ldots,N-1 \right\} \ . \tag{15.5}$$

The quantity S is a scalar that indicates the presence of the desired features. For flows with vortices, the swirl parameter, as defined in Equation 15.4, can be used as the representative scalar. The significance map delineates ROIs in the vector field of size $N \times N$ at a single resolution, usually the finest. However, the wavelet-coefficient pyramid

$$W_L = \left\{ w_k\left(i,j\right) | k = 0,\ldots,L-1,\ \ i,j = 0,\ldots,N-1 \right\} \tag{15.6}$$

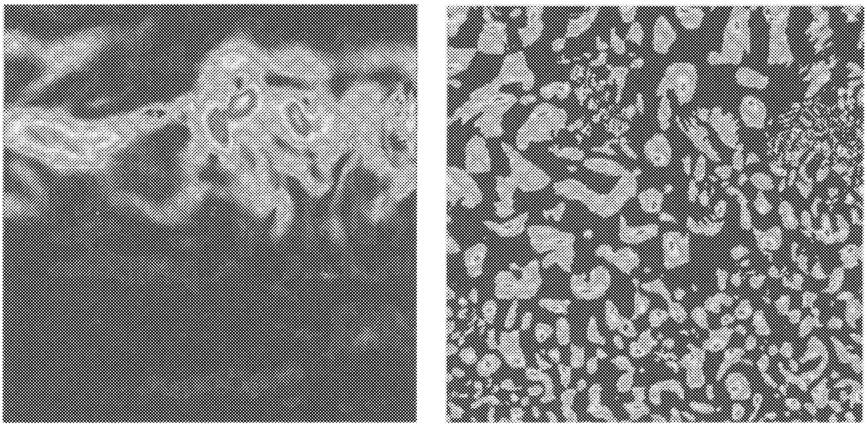

(a) (b)

Figure 2. (a) Original velocity field (magnitude) (b) swirl field.

obtained from the L-level wavelet transform of an $N \times N$-sized vector field using the lifting scheme is arranged at multiple resolutions (multi-scale). Thus, the single resolution significance map S_{single} cannot be used directly to rank the wavelet coefficients $w_k(i,j)$. This section focuses on the generation of a multi-scale significance map,

$$S_L = \{s_k(i,j) \,|k = 0 \ldots L-1, \ \ i,j = 0 \ldots N-1\} \tag{15.7}$$

that can be used directly on the multi-scale wavelet-coefficient mask W_L to determine the significance of the wavelet coefficients in terms of their contribution to feature preservation, thereby allowing ranked access to features.

Approximate Reconstruction of Features (Swirl). To generate a multi-scale significance map, it is necessary to perform a feature detection at each scale. The swirl-detection algorithm described previously requires the computation of the velocity as well as the velocity gradient at each point in the domain. Therefore, it is necessary to compute approximations to these quantities using only the local, compressed representation of the data.

A new method for approximately reconstructing the velocity and the velocity gradient using only data in the wavelet domain is now described. This technique corrects for the effects of the linear-lifting scheme as well as truncation error and is based on techniques used in multigrid-solution algorithms [Bra73]. We note that the following discussion is conducted in terms of one-dimensional data and operations. However, since a multidimensional wavelet transform is constructed as multiple one-dimensional transforms applied along each dimension in sequence, we apply the operations we describe below for one dimension in this same fashion to construct an equilavent multidimensional operator.

Let the wavelet approximation obtained after an n-level wavelet transformation of a scalar quantity $u_{j,l}$ defined on the grid $x_{j,l}$ be represented by $u_{j-n,l}$. The corresponding coarser grid is $x_{j-n,l}$ and $\Delta x_{j-n} = 2^n \Delta x_j$. Here j represents the level of the wavelet transform and l indicates the spatial position. Consider first the correction needed to account for the difference in truncation error in finite-difference approximations of the gradient using the same data on two different levels.

A second-order, central difference approximation to the gradient of u on mesh $x_{j,l}$ is given by

$$\left.\frac{du}{dx}\right|_{x_{j,2l}} = \frac{\delta u_{j,2l}}{2\Delta x_j} + \left.\frac{\Delta x_j^2}{6}\frac{d^3u}{dx^3}\right|_{x_{j,2l}} + O\left(\Delta x_j^4\right) \tag{15.8}$$

where the central difference operator is defined using

$$\delta u_{j,2l} = u_{j,2l+1} - u_{j,2l-1} \ . \tag{15.9}$$

This approximation is consistent with most CFD solutions which typically have second-order spatial accuracy. Now subsample the same data on mesh $x_{j-1,l}$ to obtain

$$\left.\frac{du}{dx}\right|_{x_{j-1,l}} = \frac{\delta u_{j-1,l}}{2\Delta x_{j-1}} + \left.\frac{\Delta x_{j-1}^2}{6}\frac{d^3u}{dx^3}\right|_{x_{j-1,l}} + O\left(\Delta x_{j-1}^4\right) \ . \tag{15.10}$$

Although the difference approximations are of the same formal order of accuracy, there is a difference in the truncation error due to the difference in the mesh spacing. This suggests adding a correction term to improve the accuracy of gradients computed on the coarser meshes. The following result is obtained using Equations 15.8 and 15.10 while generalizing to n levels noting that $\Delta x_{j-n} = 2^n \Delta x_j$:

$$\frac{\delta u_{j,2^n l}}{2\Delta x_j} = \frac{\delta u_{j-n,l}}{2\Delta x_{j-n}} - \frac{1}{6}\left(\frac{4^n - 1}{4^n}\right)\frac{d^3u}{dx^3}\Delta x_{j-n}^2 + O\left(\Delta x_{j-n}^4\right) \ . \tag{15.11}$$

Equation 15.11 specifies the correction that should be added to the central-difference approximation on level $j - n$ so that it is equivalent to the central-difference approximation on level j to the order of the approximation. A standard second-order, central-difference can be used to approximate the third-derivative term [Nak99].

Now consider the correction to the transformed quantity u needed to remove the effects of the linear-lifting scheme. Substituting the definition of the detail coefficients in the update step [Nak99] yields:

$$u_{j-1,l} = \frac{1}{8}\left(-u_{j,2l+2} + 2u_{j,2l+1} + 6u_{j,2l} + 2u_{j,2l-1} - u_{j,2l-2}\right) \ . \tag{15.12}$$

Using a Taylor series to expand Equation 15.12 about $x_{j,2l}$, yields:

$$u_{j,2l} = \left(1 + \frac{\Delta x_{j-1}^2}{16} \frac{d^2}{dx^2} + O\left(\Delta x_{j-1}^4\right) \right) u_{j-1,l} \; . \tag{15.13}$$

Through recursive application of Equation 15.13 and again noting that with $\Delta x_{j-n} = 2^n \Delta x_j$, a result generalized to n levels may be obtained:

$$u_{j,2^n l} = \left(1 + \frac{\Delta x_{j-n}^2}{16} \left(\sum_{k=0}^{n-1} \frac{1}{4^k} \right) \frac{d^2}{dx^2} + O\left(\Delta x_{j-n}^4\right) \right) u_{j-n,l} \; . \tag{15.14}$$

Equation 15.14 represents an approximation to the data on level j computed using only quantities defined on level $j-n$. Note, however, that this expression also acts as a filtering mechanism because only frequencies that can be resolved by the mesh on level $j-n$ are included in the approximation.

To complete the process, a correction must be made to the second derivative approximation similar to the one given above for the first derivative. The resulting expression is given by

$$u_{j,2^n l} = \left(1 + \frac{1}{16} \left(\sum_{k=0}^{n-1} \frac{1}{4^k} \right) \bar{\delta} + O\left(\Delta x_{j-n}^4\right) \right) u_{j-n,l} \tag{15.15}$$

where the operator $\bar{\delta}$ is defined by

$$\bar{\delta} = \delta^2 - \frac{1}{12} \left(\frac{4^n - 1}{4^n} \right) \delta^4 \tag{15.16}$$

and δ^2 and δ^4 are standard second and fourth difference operators given by

$$\delta^2 u_{j,l} = u_{j,l+1} - 2u_{j,l} + u_{j,l-1} \tag{15.17}$$

and

$$\delta^4 u_{j,l} = u_{j,l+2} - 4u_{j,l+1} + 6u_{j,l} - 4u_{j,l-1} + u_{j,l-2} \tag{15.18}$$

respectively.

The resulting procedure to compute approximations to the velocity and velocity gradients using only transformed data is as follows:

- Compute an approximate reconstruction to the velocity components at the current level using Equation 15.15 applied for each direction.
- Compute the gradients using these reconstructed values using Equation 15.11.

- Use these reconstructed values and gradient approximations to compute the swirl.

Current work includes design of new wavelet transforms that better preserve the nature of the physical simulations [TMCJ00]. Additionally, features other than swirling motion are being considered for detection and ranking.

Denoising the Data and Significance Map. The presence of oscillations in the original data can result in detection of spurious features during significance-map generation. Laplacian smoothing can be used to denoise prior to the generation of a significance map [Nak99]. Noise in the significance map S_k itself is also removed based on the persistence of features across scales.

Segmentation and Ranking of ROIs. The segmentation process involves identifying different swirl regions in the significance map and assigning each a label to facilitate later access. Here, a feature is defined as a set of connected points that have non-zero swirl and are surrounded by a zero-swirl region. A simple two-pass scanline algorithm similar to those used in image processing [Jai89] segments the features in the significance map into ROIs.

The segmented significance map identifies distinct regions as ROIs but does not indicate if one ROI is more important than another. A separate process must rank ROIs in terms of relative significance. To date, we have explored three criteria for ranking ROIs, ranking based on 1) the size of the ROI, on 2) the average strength of the feature within the ROI, and on 3) the persistence of the feature within the ROI across scales. The size of an ROI may be measured as the number of grid points contained in the ROI, while the feature strength can be calculated as the sum of the swirl values within the ROI. A straightforward ranking could be based directly on ROI size and strength as determined from a segmented, single-resolution significance map [Nak99].

A more sophisticated approach towards ROI ranking would assign an ROI containing a feature that persists across a larger number of scales a higher rank. To quantify the persistence of a feature across scales, it is necessary to track the feature across scales within a multiscale significance map. A simple tracking can be conducted by identifying features in each scale individually and then checking the spatial location of each—if the spatial locations of two features identified at different scales significantly overlap, then it is likely these features are actually the same feature being manifest at different scales. An algorithm for such scale-based ROI ranking was proposed in [Nak99]. This technique allows features to appear and disappear as the scales are traversed, and, at the end of the tracking process, features that exist from coarsest scale onwards receive a higher rank compared to those that are present only at finer scales.

In the following section we describe how the ranking of ROIs is used to influence a priority schedule among the wavelet coefficients obtained from the various ROIs. This process facilitates the realization of embedded and progressive access for large datasets.

4. Coding

The EVITA system employs state-of-the-art compression techniques that have originated in the image- and video-coding communities. In order to support user-defined priority-based access to ROIs within the compressed dataset, the design methodology underlying the EVITA system is a combination of *embedded coding* [SP96, MGBB00] and *object-based coding* [LL00b].

Embedded coders produce compressed representations at a variety of amounts of compression, where each of these representations are included ("embedded") at the beginning of a single output bitstream. That is, any prefix of an embedded bitstream can be decoded to produce a valid compressed representation of the original dataset; the longer the prefix, the more accurate is the representation. The fundamental tenant of embedded coding is therefore to put the most important information first in the encoded output. In the EVITA system, the segmented significance map is used in conjunction with user input to dynamically determine a current priority schedule governing the order of occurrence of information in the encoded bitstream.

In the EVITA system, embedded coding ranks information *within* a single ROI; to permit rankings *among* multiple ROIs, the EVITA system employs object-based coding. The philosophy underlying object-based coding is that contextually salient portions of a dataset are coded individually, so that decoding can be restricted to desired portions only. In object-based coding, this random access is supported from a single encoding, without knowledge of which portions are desired in advance of coding or the need for multiple encodings as the user's interests change.

In general terms, the compression procedure within the EVITA system follows the modern embedded-coding paradigm of wavelet transform, successive-approximation quantization, and entropy coding. Specifically, the EVITA system extends the coder developed in [FF00] for the coding of three-dimensional ocean-temperature datasets. Below, we describe some salient aspects of the EVITA encoding and decoding algorithms.

Shape-Adaptive Wavelet Transform. In the EVITA system, each ROI is encoded individually by applying a wavelet transform, successive-approximation quantization, and entropy coding only to the information in the dataset belonging to that ROI. This ROI-based coding is somewhat more complicated than traditional compression algorithms since ROIs may be arbitrarily shaped in three-dimensional space, and this shape may evolve over time (i.e., an arbitrary "shape" in four-dimensional time-space), whereas traditional compression methods are typically restricted to rectangular shapes.

In the EVITA system, individual coding of ROIs starts with a shape-adaptive wavelet transform (SAWT), such as that recently adopted by the MPEG-4 standard [LL00a]. Essentially, a SAWT is identical to a usual discrete wavelet transform except at the boundaries of an arbitrarily-shaped ROI, where a symmetric extension adapted to the shape of the ROI is employed. In the EVITA system, we use a lifting implementation of a SAWT as described in [FF00]; this transform is quite similar to the filter-bank implementation specified by MPEG-4 and described in [LL00a].

For vector-valued data fields, the transform of [FF00] can be applied to each vector component individually. Alternatively, multiwavelets [GHM94] can be used to construct more general vector-valued wavelet transforms (VWTs) [XS96] which can also be made shape-adaptive. Whether the latter approach can offer performance superior to that of the former is an open question and is a focus of our current investigations.

Successive-Approximation Quantization. For scalar sources, the EVITA system employs the successive-approximation runlength (SARL) coding technique described in [FF00]. SARL consists of "bitplane coding," wherein each wavelet coefficient is compared to a threshold, a bit is output to indicate whether the coefficient is greater than the threshold (i.e., a significant coefficient) or not (i.e., an insignificant coefficient), the threshold is halved, and the process is repeated. Thus, coefficients of an ROI are coded in a series of successive "bitplanes," starting with the most-significant bits of all coefficients in the ROI and progressing to the lesser-significant bits in order. For efficient representation, groups of successive insignificant coefficients are coded together using runlengths, and then the entire symbol stream is subjected to multiple-context adaptive arithmetic coding. SARL is particularly well suited to use in the EVITA system since it is computationally quite simple, easily deployed regardless of the dimensionality of the data, and scales well as dataset size increases (the memory usage of the "significance lists" of many popular wavelet-based algorithms such as [SP96] may quickly grow prohibitive as the size of datasets increases).

As with the wavelet transform, the straightforward solution for the successive-approximation coding of vector-valued data is to apply SARL individually to each vector component. However, more general vector-quantization (VQ) equivalents of scalar successive-approximation quantization do exist—multistage VQ (MSVQ) in particular, has been proposed frequently (e.g., [dSG95]) for successive-approximation VQ (SAVQ). We are currently exploring performance of this and other SAVQ coders within the SARL framework described above.

Transcoding. Whereas the preprocessor produces an efficient embedded encoding of each ROI suitable for initial preview representations, the server is critical to the online, interactive exploration of the dataset because it is within the server that the ROI processing takes place. The key component of the server is the *transcoder.* In general, a transcoder is a processor that converts, or translates, data encoded in one encoding into another encoding. The transcoder in our system is designed to minimize computation in order to maximize interactive response; as a result, the transcoder essentially implements a mere reordering of bits present in the compressed bitstream. [1]

As mentioned above, the successive-approximation coding of wavelet coefficients is composed of a series of bitplanes for each ROI. In the encoder in the preprocessor, the bitplanes of all the ROIs in the dataset are "interleaved" in some order and stored in an output bitstream. The order of the bitplanes in the bitstream is dictated by the relative priorities among the ROIs as specified by the significance map; this "default" ordering is intended to provide a suitable "preview" visualization if decoded as is. However, the location within the bitstream of each bitplane of each ROI is noted and output as a priority schedule by the encoder.

When the interactive visualization process commences, the transcoder initially transmits each bitplane to the client in the same order as the bitplanes are found in the compressed bitstream constructed previously by the encoder. However, when the server receives a request from the user to change the current ROI ranking, the transcoder uses the priority-schedule information to quickly locate the bitplanes of the newly selected ROI(s) so that they may be transmitted to the client immediately (or as soon as the new priority schedule dictates). Implementation of the view frustum essentially follows the same procedure as ROI selection; that is, ROIs outside the current view frustum are given the smallest priority, and a change in view frustum moves bitplanes of newly revealed ROIs to the start of the priority list.

Decoder. In essence, the decoder in the client of the EVITA system serves to reverse the operations performed by the transcoder in the server and the embedded encoder in the preprocessor. The goal of the decoder is to extract wavelet coefficients for each ROI from the received bitstream, to reconstruct each ROI via inverse quantization and inverse transform, and finally to "composite" the ROIs together into a representation of the dataset. Since ROI and view-frustum information originates at the client by way of user interaction, the decoder knows at any given time the ROI priority schedule under which the transcoder is operating. Consequently, the ordering of the bitplanes in the received bitstream is known to the decoder at all times.

---

[1] A similar process is proposed in the JPEG-2000 standard [MGBB00], wherein it is referred to as "parsing."

The decoder interacts with the visualization subsystem by producing frequent reconstructions of the dataset throughout the time that information is being received from the server. In the current implementation of the system, the decoder performs the reconstruction process after each bitplane is received from the server, although other arrangements are possible.

Grid Coding. Although the above discussion focuses on the embedded encoding of wavelet coefficients from the data field, an identical approach can be applied directly to curvilinear grid points. That is, the point locations of a curvilinear grid (an array of three-dimensional vectors) are treated just as if they were a vector-valued data field—the grid is subjected to the SAWT and SARL coding procedures described above. Then bitplanes of grid information are "interleaved" with the bitplanes of data-field information and output in the embedded bitstream produced by the encoder.

It is an open question as to how "fast" the grid information needs to be transmitted to the client. That is, the server sends bitplanes to the client to refine both the data field and grid simultaneously. Because the number of grid points is the same as the number of data-field points, the simplest approach would to refine both the grid and the data equally fast; i.e., to interleave data and grid bitplanes in a one-to-one fashion. However, because at least some crude representation of a large number (perhaps all) of the grid points may be necessary to make even an initial preview representation possible, we suspect that the best interleaving will need to include a large number of grid bitplanes initially, delaying many data-field bitplanes to later in the stream. Our initial investigations have confirmed this suspicion and seem to indicate that a large majority of the initial bits transmitted to the client need to be grid, but, after some initial time, a one-to-one mixture is warranted. We are currently devising and testing algorithms to determine optimal interleaving mixtures for data and grid dynamically as part of the transcoding operation.

Time-Varying Datasets. Time-varying datasets will be coded by merely considering time as a "fourth dimension" and employing a four-dimensional SAWT. The SARL-coding procedure generalizes to four-dimensional data trivially, while the transcoder is unaffected by the dimensionality of the data.

5. Visualization Interfaces

The hallmark of a useful interactive system is consistent, reliable response to user input. This is a particularly daunting challenge in tera-scale visualization. In order to achieve this requirement, the user interface will be designed with asynchronous communication mechanisms and an interrupt-driven event loop. Priority-driven incremental rendering (based on feature rank) will be supported and user input will be processed at a continuous rate, with interruption of the rendering process possible at any time. Caching of graphics primitives will allow information to be delivered incrementally. The cache size will be specified at runtime to support graphics platforms of different capabilities.

Additionally, the user interface must support navigation schemes and standard visualization operations such as isosurfacing, streamline generation, cut or probe surfaces, and color mapping. Navigation schemes will enable:

- traversal in three-dimensional space (gaze-directed frustum with clipping planes), time (e.g., linear scrollbar), and feature space (scalar- and vector-field parameters)
- identification and specification of ROI using mouse clicks and picking operations
- automated generation of camera paths and traversal in three-dimensional space, time, and feature space, as well as identification and specification of ROI, including automated generation of camera paths and animations
- facilities for keyframing the evolution of ROI in space and time to enable offline, higher-resolution generation of images and animations.

Information about neighboring regions will be displayed to facilitate appropriate navigation of the domain of interest. The displayed information can include statistics about number or density of feature-rich ROIs in various subvolumes along various directions relative to the current position. Such information can be determined by the preprocessor.

Finally, given the nature of ranked access of features, the system will support parceling of the original grid and transmission of the data field simultaneously with its corresponding grid points. The user can stop this progressive transmission at any time, or, as the progressive access continues across scales, the user can demand higher fidelity of either field or grid representation.

We plan to use vtk [SML98] for its extensive visualization capabilities and engineer the user interface to exploit its interactive capabilities. In the current system, only a primitive user interface exists for one to view the results of feature detection and progressive access of the data.

6. Preliminary Results

We now illustrate the EVITA system through a two-dimensional example. Figures 3 and 4 depict a visualization of a two-dimensional image of a simulated turbulent region in the Pacific Ocean [Wal99] that was obtained from an analysis of the velocity field. This region roughly corresponds to the upper left quadrant of Figures 2(a) and (b). The renderings are the scalar swirl parameter [BT93] mapped into a grayscale image. The swirl parameter is a measure of bulk

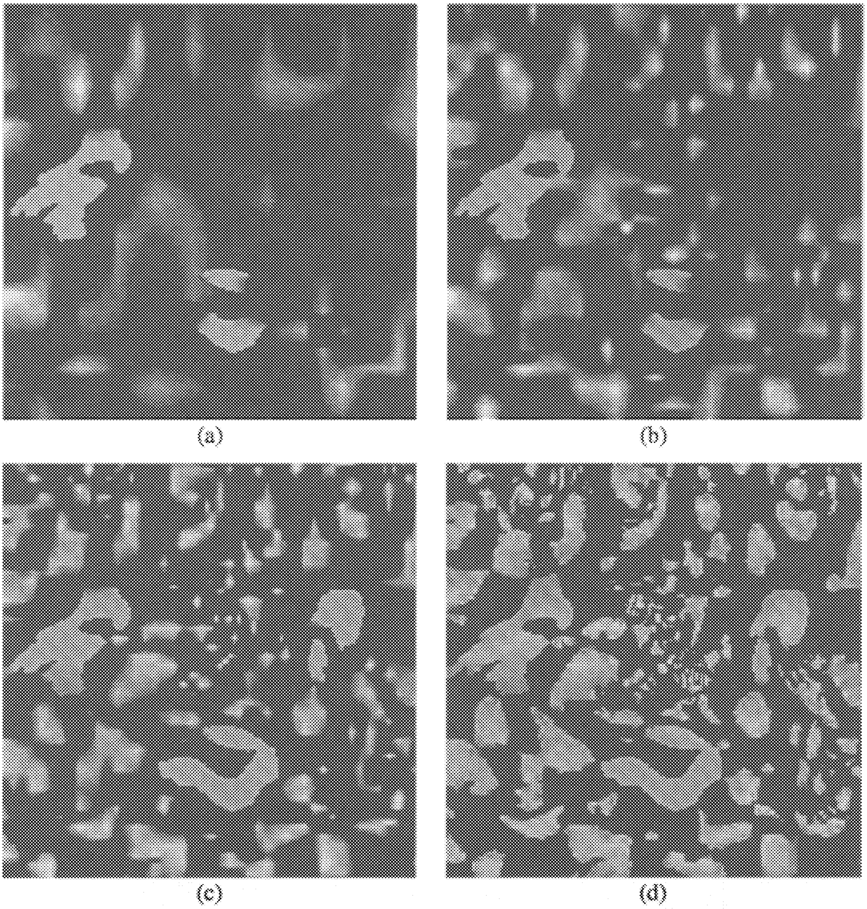
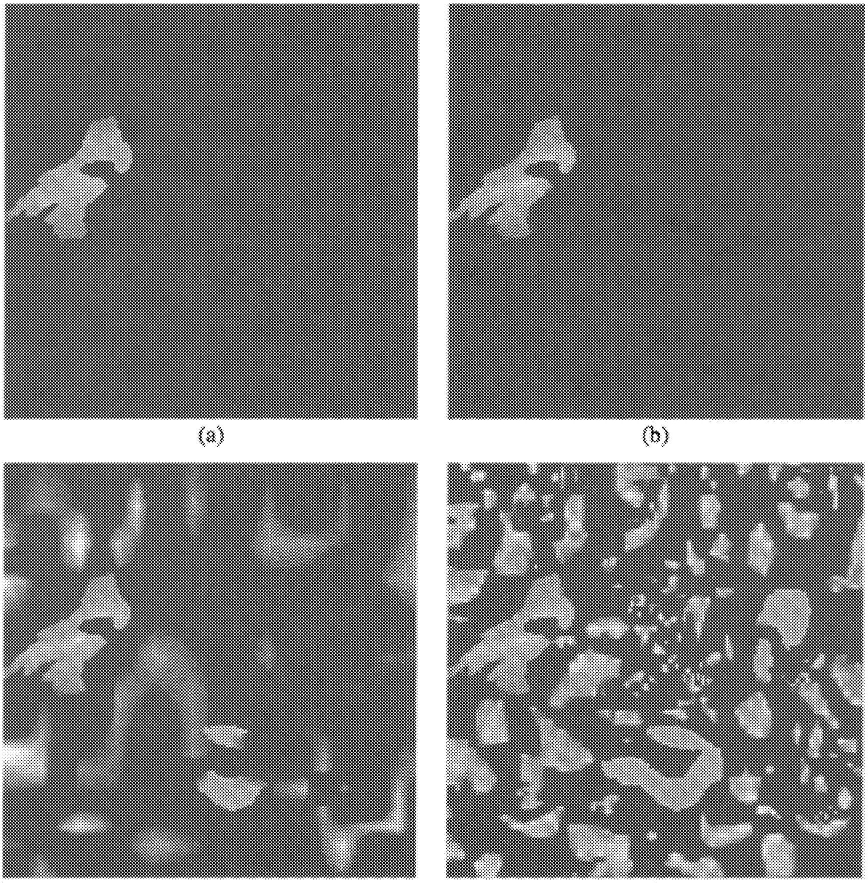


Figure 3. Pacific dataset visualization—default bitplane order as output from encoder. (a) 5th frame, (b) 9th frame, (c) 16th frame, (d) 24th frame.

rotation of fluid with a "large" value of the parameter indicating a highly rotational region; white in Figures 3 and 4 indicates "large" swirl value.

Figure 3 shows several images from the sequence of images produced by the visualization of the reconstructed swirl data using the "default" order of the bitplanes, that is, the order of the bitplanes as output originally from the offline encoder. In this default bitplane order, a bitplane is output for each ROI (including the background "ROI") in turn, resulting in refinement of all portions of the dataset equally as fast.

On the other hand, Figure 4 shows the effect on the visualization when one of the ROIs is selected by the user and given priority. In this simple example, the user selects the particular ROI at the start of the visualization process, and all bitplanes of the selected ROI are given priority over all other bitplanes (i.e.,



(a) (b)

(c) (d)

Figure 4. Pacific dataset visualization—ROI selected by user at start of visualization process and all bitplanes of selected ROI sent before rest of dataset. (a) 5th frame, (b) 9th frame, (c) 16th frame, (d) 24th frame.

the entire ROI is sent before the rest of the dataset); however, the EVITA system will support ROI selection at any time throughout the visualization session, and the priority increase given to the bitplanes of the ROI may be more gradual. In any event, it can be seen in Figure 4 that, after an ROI is selected, the system produces a quick refinement of the desired ROI, while refinement of the rest of the dataset is delayed. Regardless of the priority schedule in effect at the initial stages of the process, the visualization produced when all of the dataset is received is identical to the visualization of the full dataset.

7. Summary

Large data visualization is an ominous task. We present here a possible methodology to facilitate the exploration of large datasets—the EVITA system. EVITA was conceived to allow the continuous exploration of large datasets generated from simulations of complex physical phenomenon. Additionally, it will provide a compressed representation of both scalar and vector fields and the underlying grid that describes the physical phenomenon. To achieve both goals, a preprocessing stage constructs an embedded bitstream that includes the most significant data first. A client-server model is used to allow a user to interactively explore a large dataset, so that the client receives from the server only chosen sub-domains of the dataset which, when reconstructed in a streaming manner, allows viable visualization.

Our solution requires the generation of a four-dimensional significance map. Ideally, the significance map will segment the spatial and temporal domains into disjoint regions, or ROIs, that are tracked in time. Additionally, a ranking resulting from multiscale feature detection is obtained among the various ROIs to allow progressive access. In this feature-detection process, a wavelet transform creates a multiresolution version of the field, and the field at the coarsest level is used to measure the strength of specific features. After a process of denoising and segmentation, ROIs are delineated and a final ranking is obtained by either determining the total area and strength of the ROI, or by observing the persistence of features in ROIs across scales. As an example of the feature-detection process, we described here how bulk, macroscopic swirling motion can be located and measured as a basis for feature identification.

To achieve compression, data in each of ROI is subjected to a wavelet transform tailored to arbitrarily-shaped regions. Afterwards, a simple, yet effective embedded coding is performed according to a priority schedule as dictated by the ROI ranking. This embedded coding generates a bitstream that is amenable to partial reconstruction. Interactive exploration results from a client-server system, in which a transcoder (the server) reorders the flow of information to respond to the user's preferential interest in certain domains, while a decoder (the client) generates frequent reconstructions of the dataset as it obtains partial information from the server. In addition, the grid can be coded and transmitted in the same way using similar techniques. In the current system, a simple interface allows for the display of vector fields and rudimentary exploration.

Our preliminary results demonstrate the effectiveness of the EVITA system for a two-dimensional vector field depicting currents in the Pacific Ocean. Sample visualizations illustrate that the user determines the course of the visualization session as it is underway by selecting ROIs as desired. As a final note, we observe that, although this project is concerned primarily with feature detection and coding of fields to aid visualization, many of the techniques employed seek correlation in both spatial and temporal domains. Thus, in a broad sense, these techniques mine features and present them so as to facilitate the user's understanding of a complex physical phenomenon, which is the primary goal of traditional data-mining exercises.

Acknowledgments

This work was funded in part by the National Science Foundation under the Large Data and Scientific Software Visualization Program (ACI-9982344), and through an NSF CAREER Award (ACI-9734483). Additional support was provided by the Engineering Research Center at Mississippi State University. The authors thank B. Nakshatrala for contributions included in Section 3.

References

[Bra73] A. Brandt. Multi–level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In H. Cabannes and R. Teman, editors, *Proc. Third Intern. Conf. Num. Methods Fluid Mech.*, pages 82–89, Berlin, 1973. Springer-Verlag.

[BT93] C. H. Berdahl and D. S. Thompson. Eduction of Swirling Structure using the Velocity Gradient Tensor. *AIAA J.*, 31(1):97–103, January 1993.

[dSG95] Eduardo A. B. da Silva, Demetrios G. Sampson, and Mohammad Ghanbari. Image Coding Using Successive Approximation Wavelet Vector Quantization. In *Proc. ICASSP*, pages 2201–2204, Detroit, Michigan, May 1995.

[DTZ$^+$97] F. Davoudzadeh, L. Taylor, W. Zierke, J. Dreyer, H. McDonald, and D. Whitfield. Coupled Navier-Stokes and Equations of Motion Simulation of Submarine Maneuvers, Including Crashback. In *FEDSM97-3129, ASME Fluids Engineering Division Summer Meeting*, Vancouver, B.C., Canada, June 1997.

[FF00] James E. Fowler and Daniel N. Fox. Wavelet-Based Coding of Three-Dimensional Oceanographic Images Around Land Masses. In *Proc. Intern. Conf. Image Proc.*, pages 431–434, Vancouver, Canada, September 2000.

[Fur86] G. W. Furnas. Generalized Fisheye Views. In *Proceedings ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, Pasadena, CA., 1986.

[GHM94] Jeffrey S. Geronimo, Douglas P. Hardin, and Peter R. Massopust. Fractal Functions and Wavelet Expansions Based on Several

Scaling Functions. *J. Approx. Theory*, 78(3):373–401, September 1994.

[GSS99] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. *Computer Graphics Proc. (SIGGRAPH 99)*, pages 325–334, 1999.

[HH89] J. Helman and L. Hesselink. Representation and Display of Vector Field Topoloy in Fluid Flow Datasets. *IEEE Trans. Computers*, 22:27–36, August 1989.

[Jai89] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[Jam92] A. Jameson. Computational Algorithms for Aerodynamic Analysis and Design. Technical Report 25th Anniversary Conference on Computer Science and Control, INRIA, Paris, 1992.

[JS98] D. F. Jerding and J. T. Stasko. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Trans. Graphics Visualization*, 4(3), July-September 1998.

[KP97] L. Kantha and S. Piacsek. Computational Ocean Modeling. In *The Computer Science and Engineering Handbook*, pages 934–958. CRC Press, 1997.

[LL00a] Shipeng Li and Weiping Li. Shape-Adaptive Discrete Wavelet Transforms for Arbitrary Shaped Visual Object Coding. *IEEE Trans. Circuits Systems Video Techn.*, 10(5):725–743, August 2000.

[LL00b] Gang Lin and Ze-Min Liu. The Application of Multiwavelet Transform to Image Coding. *IEEE Trans. Image Proc.*, 9(2):270–273, February 2000.

[MGBB00] Michael W. Marcellin, Michael J. Gormish, Ali Bilgin, and Martin P. Boliek. An Overview of JPEG-2000. In James A. Storer and Martin Cohn, editors, *Proc. IEEE Data Compression Conf.*, pages 523–541, Snowbird,UT, March 2000.

[Nak99] B. Nakshatrala. Feature-Based Embedded Representation of Vector Fields. Master's thesis, Mississippi State University, December 1999.

[RCM93] G. G. Robertson, S. K. Card, and J. D. Mackinlay. Information Visualization Using 3D Interactive Animation. *Comm. ACM*, 36(4):57–71, April 1993.

[SG95] J. S. Shang and D. Gaitonde. Scattered Electromagnetic Field of a Reentry Vehicle. *J. Spacecraft Rockets*, 32(2):294–301, March-April 1995.

[SH95] D. Sujudi and R. Haimes. Identification of Swirling Flow in 3D Vector Fields. In *AIAA Paper 95-1715, AIAA 12th Computational Fluid Dynamics Conference*, San Diego, CA, June 1995.

[SML98] W. J. Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object Orient Approach to 3D Graphics.* Prentice-Hall, Englewood Cliffs, NJ, 2^{nd} edition, 1998.

[SP96] Amir Said and William A. Pearlman. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Trans. Circuits Systems Video Techn.*, 6(3):243–250, June 1996.

[SvR98] P. H. Smith and J. van Rosendale. Data and Visualization Corridors. Report on the 1998 DVC Workshop Series, Center of Advanced Computing Research, California Institute of Technology, Pasadena, CA., 1998.

[Swe97] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–526, 1997.

[TMCJ00] D. Thompson, R. Machiraju, G. Craciun, and M. Jiang. Feature Preservation Properties of Linear Wavelet Transforms (submitted). In *Proc. Yosemite Multiscale Multiresolution Symp.*, Yosemite, CA, 2000.

[Wal99] A. J. Wallcraft. The Naval Layered Ocean Model Users Guide. Technical Report 35, Naval Oceanographic and Atmospheric Research Laboratory, December 1999.

[XS96] Xiang-Gen Xia and Bruce W. Suter. Vector-Valued Wavelets and Vector Filter Banks. *IEEE Trans. Signal Proc.*, 44(3):508–518, March 1996.

Chapter 16

TOWARDS UBIQUITOUS MINING OF DISTRIBUTED DATA

Hillol Kargupta, Krishnamoorthy Sivakumar, Weiyun Huang, Rajeev Ayyagari, Rong Chen, Byung-Hoon Park, and Erik Johnson

Abstract The demand for understanding and exploring large quantity of data is growing fast in many domains. Scientific research is one among them. While the role of high performance computers in scientific data analysis is important, networks of workstations and the so called "thin" computing devices like the laptops, palmtops, and wearable computers are playing increasingly important roles in this domain. This chapter presents an overview of a collection of techniques that are designed for analyzing heterogeneous data distributed over a network of different computing and storage devices. The collective data mining approach presented here, pays careful attention to the overhead of data communication in a heterogeneous network and offers the capability of ubiquitous mining from distributed data.

Keywords: Distributed data mining, principal component analysis, clustering, decision trees, Bayesian networks, multivariate regression.

1. Introduction

Information is most useful when it can be transformed into action-able knowledge. Such transformations require efficient data access, analysis, and presentation of the outcome in a timely manner. The development of large distributed environments like the Internet and corporate intranets has added a new dimension to this process—a large number of distributed sources of data that can be used for discovering knowledge. In an increasingly mobile and connected world with a large number of distributed data sources, knowledge discovery and data mining demand due attention to the cost of communication for centralizing data. Limited network bandwidth, data security, and existing organizational structure of the applications environment often contribute to this com-

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 281–306.
© 2001 *Kluwer Academic Publishers.*

munication cost. The field of Distributed Knowledge Discovery and Data Mining (DDM) studies algorithms, systems, and human-computer interaction issues for knowledge discovery applications in distributed environments for minimizing this cost.

This chapter makes a case for DDM in scientific applications and reviews a collection of recent developments in this area. It particularly focuses on the Collective approach introduced earlier (HK99; JK99; KHK$^+$00; KPHJ00). Section 2 presents the motivation behind DDM for scientific applications. Section 3 describes the DDM problem considered here and reviews existing literature. Section 4 presents the collective principal component analysis technique. Section 5 presents distributed techniques for predictive mining using decision trees and multi-variate regression. Section 6 presents unsupervised distributed clustering techniques. It also presents a distributed Bayesian learning algorithm. Finally, Section 7 concludes this chapter.

2. Motivation

Data analysis plays an important role in modern scientific explorations. Observation of physical phenomena and experiments with computer simulations often produce large volume of data. Understanding the data requires scalable techniques for analysis. When the data sets are large, scientific data analysis is often performed in large scale high performance computing facilities. However, the growing popularity of the distributed data collection architectures for scientific applications and the possibility of ubiquitous scientific computing suggest integration of coarse-grain distributed computing in the existing high performance computing infrastructures for scientific applications. DDM fits naturally in distributed data collection infrastructures and it points out that data can be accurately analyzed without downloading everything to a central location. This approach usually offers better scalability, throughput, and overall management of data. In the following we consider a scientific application domain where DDM may significantly enhance the overall performance.

Consider a remote sensing application for studying the geophysical properties of the Earth. The NASA Earth Observing System (EOS)[1] generates more than 100 gigabytes of image data per hour, which are stored, managed, and distributed by the EOS data and information system (EOSDIS). The data managed by EOSDIS are available primarily from eight EOSDIS Distributed Active Archive Centers (DAACs). Each of the eight NASA DAACs provides data pertaining to a particular earth science discipline, as illustrated in Figure 1. These DAACs collectively

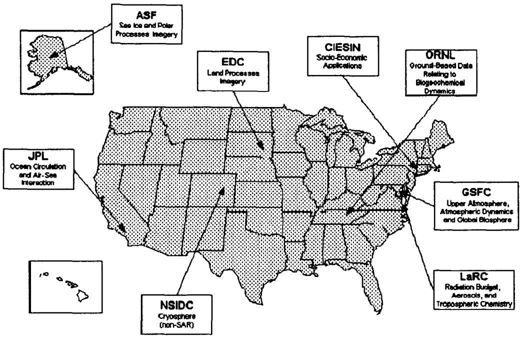

ASF
Sea Ice and Polar Processes Imagery
EDC
Land Processes Imagery
CIESIN
Socio-Economic Applications
ORNL
Ground-Based Data Relating to Biogeochemical Dynamics
JPL
Ocean Circulation and Air-Sea Interaction
GSFC
Upper Atmosphere, Atmospheric Dynamics and Global Biosphere
LaRC
Radiation Budget, Aerosols, and Tropospheric Chemistry
NSIDC
Cryosphere (non-SAR)

Figure 1. Geographical distribution of NASA DAACs.

provide a physically distributed but logically integrated database to support interdisciplinary research.

The LaRC archives, information from MISR (Multi-angle Imaging Spectro-Radiometer), CERES (Clouds and Earth's Radiant Energy System), and MOPITT (Measurements Of Pollution In The Troposphere) instruments aboard the Terra spacecraft. MISR measures top-of-atmosphere, cloud and surface angular reflectance functions, surface BRDF (bi-directional reflectance distribution function), aerosol, and vegetation MOPITT measures carbon monoxide and methane in the Troposphere. CERES measures earth's radiation budget and atmospheric radiation from the top of the atmosphere. The EDC archives, information from ASTER (Advanced Spaceborne Thermal Emission and Reflectance Radiometer) and MODIS (Moderate Resolution Imaging Spectroradiometer) instruments aboard the Terra spacecraft. MODIS provides global, near-daily repeat coverage, complementing the spectral, spatial, and temporal coverage of the other research instruments aboard the platform. ASTER data is expected to contribute to a wide array of global change-related application areas including vegetation and ecosystem dynamics, hazard monitoring, geology and soils, land surface climatology, hydrology, and land cover change. This is clearly a distributed environment for data processing. A complete usage of the gathered data and a thorough understanding of the earth's natural processes require an enormous amount of knowledge discovery from the distributed sites of this infrastructure. Doing this in a scalable and timely fashion requires a comprehensive data mining approach that pays careful attention to different aspects of distributed computing and data storage.

The distributed data collection infrastructures are also complemented by the need for distributed and ubiquitous access to data. There are many domains like disease monitoring, agriculture, and energy management, where scientific data mining practitioners may have to analyze data using mobile devices connected by limited bandwidth wired/wireless network. These applications also call for the development of a scalable framework for distributed data analysis.

DDM offers a data mining solution for distributed environments. It notes that communication of large amount of data over the network and the centralized architecture may cause poor response time and scalability. So it performs partial analysis of data at individual sites and sends out the results to other sites. Typically a small fraction of data is also moved. This subset of data is sometimes required to correctly aggregate the partial results together and to validate the resulting global outcome. DDM replaces communication of large amount of distributed data by that of tuples like, (*partial model, data subset*). This saves communication overhead, offers better scalability, requires limited communication of possibly confidential data, and sometimes offers the only feasible way to mine distributed data sets.

Development of DDM applications requires scalable techniques for distributed data analysis, appropriate system architecture, and novel techniques for distributed human-computer interaction. This chapter considers the first issue and presents a collection of DDM algorithms that can be used for distributed analysis of scientific data.

3. Background

DDM application environments may have different kinds of data distribution scenarios. Different sites may observe different events defined by the same set of features. On the other hand different sites may observe related events described by different sets of features. They are called the *homogeneous* and the *heterogeneous* cases respectively. In this chapter, we shall restrict ourselves to relational data. Therefore, we can assume that homogeneous databases have a consistent schema across all the sites. This is not the case in the heterogeneous DDM environment. In the latter case, we assume that the rows in different tables are associated through some common index keys. A detailed description can be found in (KPHJ00). This chapter considers DDM from heterogeneous sites.

The volume of DDM literature is growing fast. There exist a reasonably large body of work on DDM architectures and data mining techniques for the homogeneous and heterogeneous cases. The following part of this section reviews the existing literature for heterogeneous DDM.

Mining from heterogeneous data constitutes an important class of DDM problems. This issue is discussed in (PB95) from the perspective of inductive bias. The WoRLD system (AKPB97) addressed the problem of concept learning from heterogeneous sites by developing an "activation spreading" approach that is based on first order statistical estimation of the underlying distribution. A novel approach to learn association rules from heterogeneous tables is proposed in (CS99). This approach exploits the foreign key relationships for the case of a star schema to develop decentralized algorithms that execute concurrently on the separate tables, and subsequently merge the results. An order statistics-based technique for combining high-variance models generated from heterogeneous sites is proposed in (TG00a).

Kargupta and his colleagues (KPHJ00) also considered this case and proposed the *Collective* framework to address data analysis for heterogeneous environments. They proposed the *Collective Data Mining* (CDM) framework for predictive data modeling that makes use of orthonormal basis functions for correct local analysis. They proposed a technique for distributed decision tree construction (KPHJ00) and wavelet-based multi-variate regression (HK99). Several distributed clustering techniques based on the Collective framework are proposed elsewhere (JK99; KHKJ00). They also proposed the collective PCA technique (KHKJ00; KHK$^+$00) and its extension to a distributed clustering application. Additional work on distributed decision tree learning (BS97), clustering (MSG00; PO00; SS00), genetic learning(PKJ$^+$00) DDM design optimization (TG00b), classifier pruning (PS00), DDM architecture (KN96), and problem decomposition and local model selection in DDM (LPO00) are also reported. The following section considers a distributed representation construction technique.

4. Representation Construction

Representation plays an important role in scientific data mining. For example, a large proportion of scientific data are image data, such as cloud observations provided by satellites, remote sensing data of oceans, information about vegetation and hydrology changes. Mining such data sets often require extracting features from raw data in order to generate effective models. Principal Component Analysis (PCA) (Jac91) is good at extracting significant and independent features from high dimensional data. It is essentially a linear transformation, so it may not be suitable to some non-linear data sets. However, generally speaking, it is a popular statistical method for efficiently representing scientific data sets. Performing PCA in a distributed environment is important for many applications.

4.1. The Collective PCA

This section presents the Collective PCA (CPCA) algorithm, which can perform distributed PCA from heterogeneous data sets with limited communication of raw data. For the sake of simplicity we consider only one table per site. The technique can be easily extended to problems with multiple tables at each site.

In a distributed and heterogenous environment, the entire data matrix X is comprised of different smaller tables stored at different sites. In particular, each site has data regarding a subset of features. Let us assume that the data is distributed among s sites. A PCA of this distributed data in a centralized fashion would involve moving data to one central site This involves $O(mn)$ data communication, for a $m \times n$ global data matrix with m data samples and n features. For typical DKD-applications this amount of data communication is either prohibitive because of the limited bandwidth or impractical because of logistics and/or security related reasons.

Given the distributed nature of the data, it would be advantageous to perform the computations for PCA (to the extent possible) locally, thereby minimizing the amount of data communication and the computation at the central site. The overall CPCA algorithm is summarized in the following:

Local site: Perform local PCA at each site, select dominant eigenvectors, and project the data along them. Send a sample of the projected data along with the eigenvectors to central site.

Central site: Combine the projected data from all the sites. Perform PCA on the global data set, identify the dominant eigenvectors, and transform them back to the original space.

In the first step, we select dominant local principal components (PCs) according to a PC selection threshold. That is, for each site i, we choose k_i PCs corresponding to the k_i dominant eigenvalues of covariance matrix. The introduced relative error is essentially the ratio of the sum of k_i largest eigenvalues to the sum of all the eigenvalues. At the central site, in principle, we need to reconstruct the original data X from the sampled projections (denoted as Y_i for site i) and local PCs (denoted as A_i for site i). However, since the PCA is invariant to linear transformations (Jac91), we can work with the Y data instead of X. If $Y = [Y_1, Y_2, \ldots, Y_s]$, we have $Y = XA$, where $A = \text{diag}\{A_1, \ldots, A_s\}$, is a block diagonal matrix. We can compute dominant eigenvectors v_i, $i = 1, 2, \ldots, p$ corresponding to the p largest eigenvalues of the covariance matrix of Y. Then $w_i = Av_i$ are the required eigenvectors of the covariance $X'X$ of the original data X.

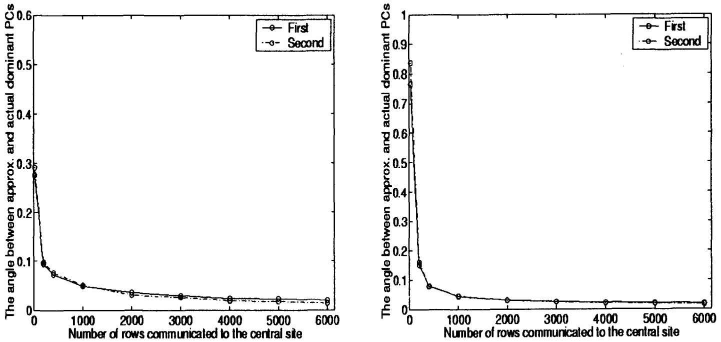


Figure 2. Performance of CPCA for the robot move data distributed among: two sites (left) and six sites (right). The graphs are averages of five independent runs. "First" and "Second" correspond to the two dominant eigenvectors.

4.2. Experiments with the CPCA

In this section we show that the CPCA achieves highly accurate results compared to the centralized approach (which performs PCA after downloading all the tables to a single site) and it does so with little communication overhead. To quantify the error, we plot the angle (in radians) between the approximation of PCs generated by CPCA and the ones obtained by a centralized PCA. Two curves in the Figures 2 represent the errors obtained for the first two dominant PCs. We use Robot Move data^2 for our experiment. It is comprised of 6129 observations and 36 features (only numeric features are used). We vertically partition the data set into two subsets (to be stored at two sites), with 18 features each.

We set the PC selection threshold to 0.90. It turns out that 6 local PCs for the first partition and 3 for the second partition are sufficient to satisfy the threshold. A random subset of the rows (same subset for all sites) and the local PCs for each partition are then sent to a central site. Next global PCA is performed on the aggregated data at the central site in order to produce the global principal components.

Figure 2 shows the performance of CPCA with respect to the number of rows that are sampled from local projections, when data is split across two sites (left) and six sites (right). The result shows that for these data sets sampling about 20% of the rows in the projected space is sufficient for highly accurate result. As to the number of selected local PCs, the two sites case selected 9 in total, and the six sites case selected 11 in total.

We see from the figures that the CPCA achieves good compression. Suppose our global data set is $m \times n$, and is distributed among s sites, such that each site contains n_i features. Number of selected local PCs is k_i, and the total number of selected local PCs is $k = \sum_{i=1}^{s} k_i$. Usually when the PC selection threshold is set to 0.90 or 0.95, k is quite small compared to n. Let us denote the number of sampled rows by c; for our experiments we chose a value of c such that $c < 0.2m$. The total communication cost is $O(ck + \sum_{i=1}^{s} n_i k_i)$, which is very small compared to $O(mn)$, the cost to move the whole data set to one site.

5. Predictive CDM

This section considers decision trees and multi-variate regression in a distributed heterogeneous environment.

5.1. Decision Trees

Decision Trees (DTs) (Qui86) have been applied to a wide variety of scientific data mining problems. Areas as diverse as DNA analysis, cosmology and acoustics have benefitted from the use of DTs (SCF$^+$95; SCD$^+$97; Col87). Distributed DT learning algorithms can be useful for mining distributed scientific data. An example is scientific mining of weather data. DTs can be used to predict thunderstorm types (Col87). Another application involves the classification of objects in astronomical images. As reported in (SCF$^+$95), oblique DTs are useful for classifying objects in astronomical images from the Wide Field Cameras (WFCs) of the Hubble Space Telescope (HST). With the growing popularity of distributed infrastructure for scientific data collection, the need for distributed DT learning techniques for such applications seems natural.

A straight-forward construction of a distributed version of the DT induction algorithms like ID3 (Qui86) produces a highly synchronous algorithm that may not scale up for large heterogeneous environments (KPHJ00). We think that many interesting applications of distributed DT learning require an loosely coupled asynchronous algorithm that can work independently at each site. This section reviews a novel Fourier analysis-based technique to do so.

A DT can be viewed as a function and therefore we can compute its Fourier Spectrum (FS). DTs with finite depths can be represented accurately using a small number of Fourier Coefficients (FCs) bounded by polynomials of ℓ, the number of features. Moreover, only a small fraction of those coefficients are exponentially more significant than the rest. Therefore, the FS of a DT can be approximated by computing only a small number of FCs (KPHJ00). In other words, the representation of a DT is quite concise and efficient in the Fourier basis. A DT can be learned by estimating these coefficients from the data. This section summarizes the collective DT learning algorithm that works by computing the coefficients in a distributed fashion, aggregating them, and finally building the global tree from the combined spectrum.

5.2. Decision Trees and the Fourier Domain

This section introduces the Fourier Spectrum of a DT and shows how to calculate the FS from the DT and vice versa. It suggests how to combine the spectra of an ensemble of DTs.

A Brief Review of the Fourier Basis. Consider the set $\mathcal{X}$ of all ℓ-bit Boolean feature vectors. (We introduce only the discrete boolean Fourier basis here. See (KPHJ00) for the non-boolean case.) The Fourier basis is an orthonormal set of functions that spans the space of all real-valued functions on $\mathcal{X}$. It is comprised of 2^ℓ functions, $\psi_{\mathbf{j}}(\mathbf{x}) = (-1)^{(\mathbf{x} \cdot \mathbf{j})}$, where $\mathbf{j}, \mathbf{x} \in \{0,1\}^\ell$. $\mathbf{x} \cdot \mathbf{j}$ denotes the inner product of $\mathbf{x}$ and $\mathbf{j}$. The string $\mathbf{j}$ is called a *partition*. The *order* of a partition $\mathbf{j}$ is the number of 1-s in $\mathbf{j}$. A Fourier basis function depends on some x_i only when $\mathbf{j}_i = 1$. A function $f : \mathcal{X} \to \Re$, can be represented as a linear combination of the Fourier basis functions: $f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}} \psi_{\mathbf{j}}(\mathbf{x})$ where $w_{\mathbf{j}}$ is the FC corresponding to the partition $\mathbf{j}$ [3]; $w_{\mathbf{j}} = \frac{1}{2^l} \sum_{\mathbf{x}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x})$. The FC $w_{\mathbf{j}}$ represents the relative contribution of the partition $\mathbf{j}$ to the function $f(\mathbf{x})$ and can be viewed as the "significance" of that partition.

Fourier Transform of a Decision Tree. The $\mathbf{j}$-th FC $w_{\mathbf{j}}$ of a DT can be computed as follows:

$$\frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{n} \sum_{\mathbf{x} \in S_{l_i}} f(\mathbf{x}) \psi_{\mathbf{j}}(\mathbf{x}) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{n} |S_{l_i}| f(\mathbf{h}_i) \psi_{\mathbf{j}}(\mathbf{h}_i)$$

where S_{l_i} is an instance subspace which i-th leaf node l_i covers , $\mathcal{X}$ denotes the complete instance space as defined earlier and $\mathbf{h}_{\mathrm{i}}$ is a data subset that satisfy the constraints imposed by a path from the root node to l_i. $\mathbf{h}_{\mathrm{i}}$ is essentially a similarity based equivalence class and we shall call it schema.

It turns out that the FS of a DT has some very interesting properties (LMN93; KP01) which helps to detect significant FCs easily.

1 Let $\mathbf{h}_{\mathrm{i}}$ be the schema defined by the path to a leaf node l_i. Then if $\mathbf{j}$ has a non-zero bit at a position where h_i has a zero bit, $\sum_{\mathbf{x} \in S_{l_i}} \psi_{\mathbf{j}}(\mathbf{x}) = 0$, where S_{l_i} is the subset of domain that $\mathbf{h}_{\mathrm{i}}$ covers.

2 For any FC $w_{\mathbf{j}}$ whose order is greater than the depth of a leaf node l_i, $\sum_{\mathbf{x}\in S_{l_i}} \psi_{\mathbf{j}}(\mathbf{x}) = 0$. If the order of $\mathbf{j}$ is greater than the depth of the tree, then $w_{\mathbf{j}} = 0$.

3 The magnitudes of the FCs decay exponentially with the order of the corresponding partitions.

Thus, for a FC $w_{\mathbf{j}}$ to be non-zero, all the attributes with non-zero values in $\mathbf{j}$ should appear in at least one schema $\mathbf{h}$ defined by a path to a leaf node. This indicates a method to detect and calculate all non-zero FCs of a DT. For each path from the root to a leaf node (schema $\mathbf{h}$), all non-zero FCs are detected by enumerating all possible values for each attribute in $\mathbf{h}$. The number of FCs of a DT grows fast as the size of DT. However, due to the exponential decay property of FCs, we can approximate FS of a DT with only low order FCs.

Fourier Spectrum of an Ensemble Classifier. The FS of an ensemble classifier can also be obtained by applying the approach described above. Suppose $f(\mathbf{x})$ is the function representing the ensemble classifier and $f(\mathbf{x})$ is a linear combination of classifications of multiple trees:

$$f(\mathbf{x}) = \sum_{i=1}^{n} a_i f_i(\mathbf{x}) = \sum_{i=1}^{n} a_i \sum_{\mathbf{j}\in J_i} w_{\mathbf{j}}^{(i)} \psi_{\mathbf{j}}(\mathbf{x})$$

where $f_i(\mathbf{x})$ and a_i are the i-th DT and its weight respectively; J_i is the set of all non-zero FCs that are detected by i-th DT. Hence $f(\mathbf{x}) = \sum_{\mathbf{j}\in J} w_{\mathbf{j}} \psi_{\mathbf{j}}(\mathbf{x})$ where $w_{\mathbf{j}} = \sum_{i=1}^{n} a_i w_{\mathbf{j}}^{(i)}$ and $J = \cup_{i=1}^{n} J_i$. This offers a way to aggregate multiple decision trees and analytically manipulating them in the Fourier domain.

5.3. Modeling Heterogeneous Data using Decision Trees

In this section, we consider a paradigm which allows us to apply the techniques of the last section in distributed learning.

Model Building and Aggregation. The approach involves three steps.

First, boosting (Sch90; Fre95; FS97) is performed at each local site to obtain an ensemble of local DTs. Each tree outputs a real number which indicates a confidence as well as a classification. An overall confidence is calculated at each site for every training instance (a weighted average of the confidences of individual trees). Instances classified with low confidence at all sites are sent to a global site, which builds another ensemble of DTs from these. This ensemble captures the "cross terms" in the data, that is, interactions between features at different sites.

The model at this stage consists of a set of DTs at each local site and at the global site; we must aggregate them. Of several techniques (Weighted Average, Ordinary Least Squares and Tournament Based) we considered, a Tournament based scheme appeared to produce the best results. In this scheme, unseen instances are first classified using the local models. Those which are classified with low confidence by all the local models are classified using the global model. An expanded description of this scheme is available elsewhere (PAK01; PKA01). At this stage, we have a fairly complete classifier. However, such a classifier is typically not very useful from the data mining perspective. It is usually comprised of a large number of trees and the aggregation is too complicated to comprehend. The techniques used to aggregate FS of an ensemble classifier can be used to mitigate this problem by combining the DTs at each local site into a single, approximately functionally equivalent tree.

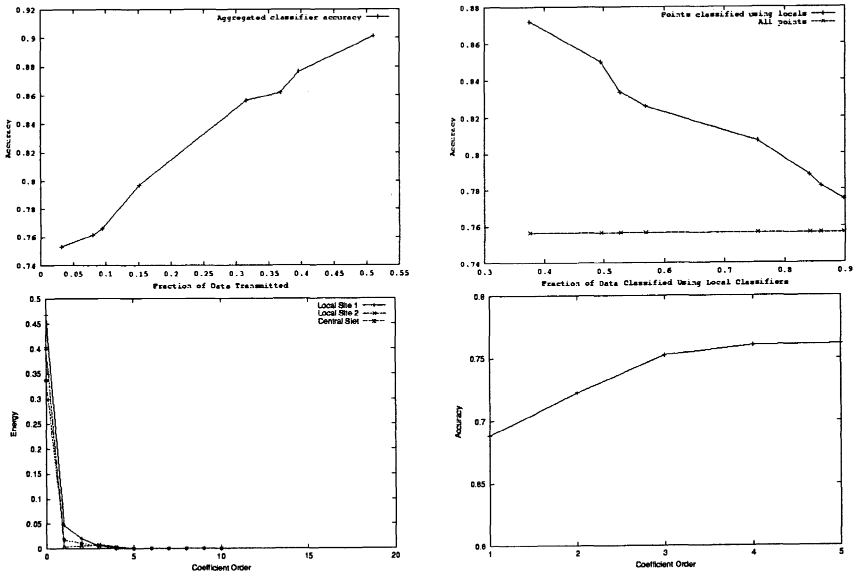


Figure 3. The Figures (left to right and top to bottom) are: (a) Accuracy vs. transmission for Gene data, (b) Accuracy of local models vs. instances classified by them, (c) Energy of coefficients vs. their order, (d) Accuracy of Fourier representation based model vs. order of coefficients used.

5.4. Experiments and Conclusions

This section describes experimental verification of the techniques presented above. The experiments were performed on Promoter Gene Sequence data, which was artificially generated using a DT built by C4.5 learning from the original Promoter Gene Sequence data set obtained from the UCI Machine Learning Repository (`http://www.ics.uci.edu/~mlearn/MLRepository.html`). The data used to build the tree had 76 instances with 57 attributes. Each attribute had 4 four possible values: A, C, G, T (Adenine, Cytosine, Guanine, Thymine). The final data set had 50,000 training and 5,000 test instances.

The results are shown in Figure 3. The dataset was split into 5 sites. Graph (a) shows the accuracy of the algorithm for various values of the transmission rate (% of data transmitted to the central site). Note that only about 55% transmission is required to achieve almost full accuracy. Graphs (b) show the accuracy versus the percentage of points classified by a the local sites alone without use of the global site. The horizontal line represents the accuracy achieved when the global site is not used at all. The readings for the graph were generated by varying the threshold value, thereby controlling the amount of data classified by the central site. Note that, compared to the decentralized (horizontal) graph, the local sites perform very well on the instances that they classify, but comparatively poorly on the set of all training instances. This shows that the local sites are good classifiers only for the instances they classify. This is indeed one of the goals of the algorithm: to identify which data points the local sites are likely to classify well, and which need to be sent to the global site. Graph (c) demonstrates the exponential decay property of the Fourier coefficients of a DT. Graph (d) shows the accuracy of the Fourier representation of the entire model when coefficients up to a certain order are used.

The results show that the proposed Fourier analysis techniques perform well. The boosting-based method for detecting the partitions which need to be estimated at the global site also seems to be working well. Thus, these techniques form a promising suite of methods for analysis of distributed data in a consistent fashion.

While the techniques presented appear to perform well, many refinements and improvements need to be made. Alternative schemes for aggregating decision trees may produce better results in certain applications. A method for combining heterogeneous DTs (that is, DTs with different feature spaces) according to the tournament scheme is currently under development.

The collective technique presented here works with other predictive learning tools as well. We indicates how multi-variate regression can be brought under the collective framework. This technique is detailed elsewhere (HK99). Regression techniques are often used in data modeling and knowledge discovery applications. Therefore, development of its distributed version for heterogeneous sites is of high interest. This section summarizes a technique for developing a wavelet-based CDM technique for multivariate polynomial regression. The overall approach is comprised of the following steps: (1) compute the wavelet transformation of the data; (2) select the significant coefficients from different sites and collect them to a single site; (3) regress directly on the wavelet coefficients; (4) construct the regression model in the canonical representation from the model in the wavelet space.

6. Unsupervised Collective Data Mining

This section presents three clustering techniques developed elsewhere (JK99; KHKJ00), and a collective Bayesian learning technique.

6.1. Distributed Clustering Using the CPCA

This section demonstrates that the CPCA technique, introduced earlier, can be easily integrated with standard off-the-shelve clustering algorithms in order to generate a distributed clustering technique.

The proposed distributed clustering approach respects the user's choice of any specific local clustering algorithm and works using a given module of centralized clustering algorithm $\mathcal{C}$. It executes the following steps:

1 Performs local PCA at each site, projects the local data on the local principal components (PCs), and applies the given clustering algorithm $\mathcal{C}$ at each site.

2 Selects a set of representative points from each cluster at every site. Let S_i be the set of indices at site i corresponding to the chosen representative points. All sites communicate the projected data rows corresponding to all indices in $\cup_i S_i$ to the central site.

3 The central site performs the global PCA on this collected data set and broadcasts the global PCs to each site.

4 Each site projects the local data on the global PCs, performs clustering using the given algorithm $\mathcal{C}$, and communicates a description of the locally constructed clusters to the central site.

5 The central site combines the different graphs obtained from the local sites.

First the local PCs are computed for compressing the local data. After that, in stead of uniform sampling the projections, we take advantage of the given local clustering technique to select a set of points that represent all the different clusters. This reduces the possibility of choosing a sample set that completely neglects some clusters present in the data set. Next the global site computes the global PCs and broadcasts them to each site.

Let V be an $n \times p$ matrix such that its columns are the global PCs, i.e., the p global eigenvectors corresponding to the p dominant eigenvalues. The projection of the complete data set X along V is XV. Unfortunately, the data set X is distributed among different sites, i.e. $X = [X_1, X_2, \ldots, X_s]$. Note that X_i (the data set at site i) has m rows and n_i columns. Moreover, we can write $V = [V_1', V_2', \ldots, V_s']'$, where V_i is an $n_i \times p$ sub-matrix. Therefore, $XV = [X_1V_1 + X_2V_2 \cdots + X_sV_s]$ and the i-th site computes only X_iV_i.

Clustering the projected data set XV typically requires computing an appropriately chosen distance metric. In our case we consider the Euclidean distance metric. Let us consider two data points $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s]$ and $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_s]$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are $1 \times n_i$ dimensional row matrices. Computing the distance between $\mathbf{x}$ and $\mathbf{y}$ requires central collection of the $\mathbf{x}_1V_1$, $\mathbf{x}_2V_2$, and $\mathbf{x}_sV_s$. However in the following we show that sum of the distances between $\mathbf{x}_iV_i$ and $\mathbf{y}_iV_i$ for all i is a good approximation of the overall distance between $\mathbf{x}$ and $\mathbf{y}$.

Define $\mathbf{x} - \mathbf{y} = \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_s]$. Let $U = [U_1', U_2', \ldots, U_s']'$, where U_i is an $n_i \times (n-p)$ sub-matrix, be the matrix whose columns correspond to the $(n-p)$ global eigenvectors corresponding to the $(n-p)$ smallest global eigenvalues. Recall that, in practice, these eigenvectors are not computed and correspond to the $(n-p)$ PCs that are ignored. It is well known that the $n \times n$ matrix $[V\ U]$ is orthogonal. In particular, its rows form an orthonormal basis.

For notational simplicity, we will consider a two site case (i.e., $s = 2$). The extension to the general case is straight-forward. Using *all* the global eigenvectors, the projection of $\mathbf{z}$ can be written as

$$\begin{aligned}\begin{bmatrix} \hat{\mathbf{z}} & \hat{\mathbf{w}} \end{bmatrix} = \mathbf{z}\begin{bmatrix} V & U \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix}\begin{bmatrix} V_1 & U_1 \\ V_2 & U_2 \end{bmatrix} \\ = \begin{bmatrix} \hat{\mathbf{z}}_1 & \hat{\mathbf{w}}_1 \end{bmatrix} + \begin{bmatrix} \hat{\mathbf{z}}_2 & \hat{\mathbf{w}}_2 \end{bmatrix}.\end{aligned}$$

The vectors $\hat{\mathbf{z}}_1 = \mathbf{z}_1V_1$ and $\hat{\mathbf{z}}_2 = \mathbf{z}_2V_2$ are the local PCs computed at sites 1 and 2, respectively, whereas the vectors $\hat{\mathbf{w}}_1 = \mathbf{z}_1U_1$ and $\hat{\mathbf{w}}_2 = \mathbf{z}_2U_2$ are the PCs that are ignored. Since the rows of matrix $[V\ U]$ form an

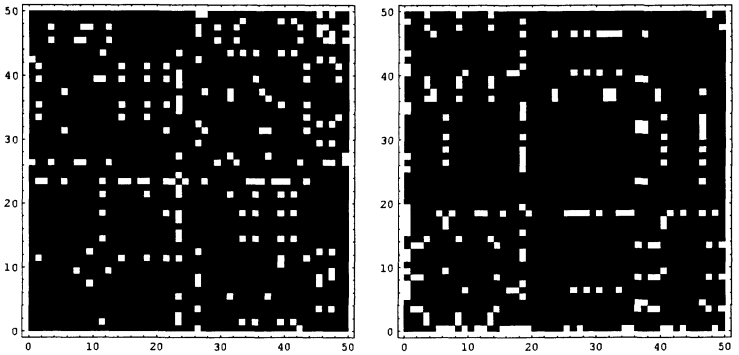


Figure 4. The difference between adjacency matrices generated by the centralized approach and the distributed approach: (Left)Data are distributed among two sites.(Right) Data are distributed among ten sites.

orthogonal basis, vectors $[\hat{\mathbf{z}}_1\ \hat{\mathbf{w}}_1]$ and $[\hat{\mathbf{z}}_2\ \hat{\mathbf{w}}_2]$ are orthogonal. Therefore,

$$\begin{aligned}\|\mathbf{x}-\mathbf{y}\|^2 = \|\mathbf{z}\|^2 = \left\|\left[\begin{array}{cc}\hat{\mathbf{z}} & \hat{\mathbf{w}}\end{array}\right]\right\|^2 &= \left\|\left[\begin{array}{cc}\hat{\mathbf{z}}_1 & \hat{\mathbf{w}}_1\end{array}\right]\right\|^2 + \left\|\left[\begin{array}{cc}\hat{\mathbf{z}}_2 & \hat{\mathbf{w}}_2\end{array}\right]\right\|^2 \\ &= \|\hat{\mathbf{z}}_1\|^2 + \|\hat{\mathbf{w}}_1\|^2 + \|\hat{\mathbf{z}}_2\|^2 + \|\hat{\mathbf{w}}_2\|^2.\end{aligned}$$

It follows that $\|\mathbf{z}\|^2 - (\|\hat{\mathbf{z}}_1\|^2 + \|\hat{\mathbf{z}}_2\|^2) = \|\hat{\mathbf{w}}_1\|^2 + \|\hat{\mathbf{w}}_2\|^2$, where $\hat{\mathbf{z}}_i = \hat{\mathbf{x}}_i - \hat{\mathbf{y}}_i = \mathbf{x}_i V_i - \mathbf{y}_i V_i$, for $i = 1, 2$.

It is now easy to see that the error committed by approximating the (squared) distance between data points $\mathbf{x}$ and $\mathbf{y}$ by the sum of the (squared) distances between the appropriate local PCs is relatively small. Indeed, it is directly related to the norm of the principal components that have been ignored, which is usually quite small.

Therefore we can approximate the global distance among a pair of points by simply computing the local distances and adding them accordingly. As a result the generated global clusters offer good approximation of the clusters that can be obtained by the corresponding centralized technique. Figure 4 shows the difference between adjacency matrices generated by the centralized clustering and the distributed clustering techniques, when data are distributed among two sites(left) and ten sites (right). In the adjacency matrix the (i, j)-th entry of the matrix is 1 if the i-th and the j-th data rows belong to the same cluster. Otherwise the value is set to 0. Then we take the XOR of two matrices to get their difference. The following section presents the collective approach for hierarchical clustering.

6.2. Collective Hierarchical Clustering

This section presents the collective hierarchical clustering technique to build a global hierarchical cluster model from distributed, heterogeneous data sites. The global model should approximate the model that would have been generated had the data been centralized, with approximately the same time requirements. The overall approach is summarized in the following.

First, local models are generated in the form of dendrograms at each of the distributed sites. Established agglomerative, hierarchical clustering algorithms are used to generate the local models. It should be noted that this scheme is independent of the specific clustering algorithm used, as long as consistency of algorithm is maintained at each of the independent sites.

The local models are then transmitted to the central, *facilitator* cite. As these local models consist of a tree structure, the transmission of the flattened tree can be accomplished with an $O(m)$ communication cost, where m is the number of rows in the table.

Following transmission, and using distance bounds for the minimum and maximum possible distance between any two points computed from the local models, a global model is generated from the locally generated models which approximates the model generated by centralized approach. The distance between any two given points, p_1 and p_2, at the facilitator site has a distinct upper and lower bound. Suppose we have some set of sites S, with cardinality $|S|$, and further uppose the data is distributed among these $|S|$ sites. At any one of the $|S|$ sites, the minimum distance between these two points in the feature space of that site will be the distance stored in the root of the minimum subtree connecting the two points. Hence, the lower bound of the distance between the two points in the overall feature space can be expressed as: $dist_{min}(p_1, p_2) = \sqrt{\sum_{j=1}^{|S|} d^2_{commonroot,j}}$ where $d_{commonroot,j}$ is the distance stored in the root of the minimum subtree at site j connecting points p_1 and p_2.

The upper bound for the distance between any two arbitrary points p_1 and p_2 can be similarly calculated from the information stored in the dendrograms. At any one of the $|S|$ sites, the maximum possible distance between any two points is the sum of all the distances stored in the connecting path through the minimum subtree connecting the two points. Assuming an Euclidean space, and applying the triangle law, the overall maximum distance between the two points at the facilitator site can be expressed as: $dist_{max}(p_1, p_2) = \sqrt{\sum_{j=1}^{|S|} \left(\sum_{shortpath,j} d_{pathnode}\right)^2}$; where *shortpath* is the set of all the internal nodes for site j in the minimum connecting subtree between points p_1 and p_2 and $d_{pathnode}$ is the distance stored in the individual internal nodes of the dendrogram. Through adaptation of the SLINK (Sib73) algorithm, this process can

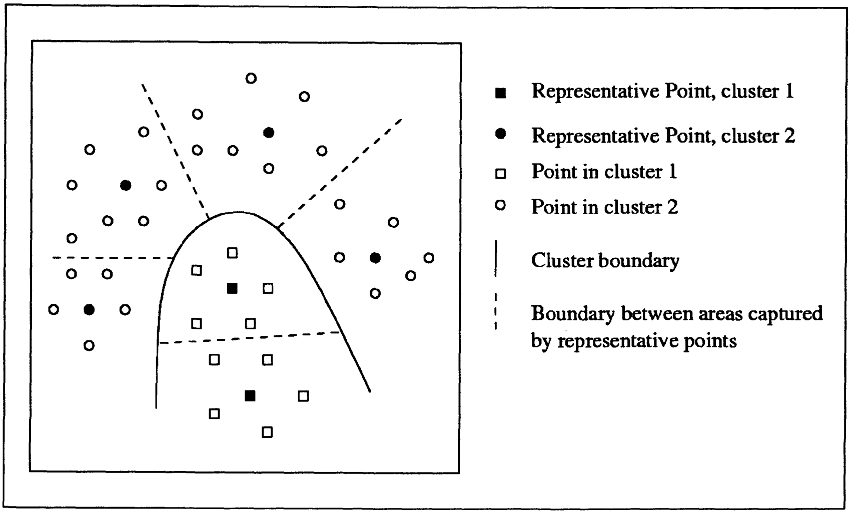

■ Representative Point, cluster 1
● Representative Point, cluster 2
□ Point in cluster 1
○ Point in cluster 2
Cluster boundary
Boundary between areas captured by representative points

Figure 5. Two clusters, and their representative points. The solid line indicates the boundary between the two clusters, and the dashed lines shows the areas of the cluster described by the single representative point contained within that portion of the cluster.

be accomplished in $O(n^2)$ time and $O(n)$ space, where n is the number of rows. For more details, including the full algorithm and its derivation, see (JK99).

Once the global model has been generated, if more detailed information is required about a given set of points, specific information concerning feature values can be requested from the local sites.

The following section presents a more general approach to collective clustering in the heterogeneous case.

6.3. Collective Meta Clustering

The Collective Meta Clustering (CMC) approach to distributed, heterogeneous clustering uses the concept of *representative points* (GRS01) to describe the shape of the space being clustered. A representative point, or *meta point*, is a point used to describe one or more points that have a proximity closer to the given representative point than to any other representative point. Figure 5 shows an example of representative points.

As with the CHC approach, the goal is to generate a global model from the local models which approximates the model generated by centralized approach. With the CMC approach, however, the overall error can be defined by the user of the algorithm. Furthermore, the CMC approach takes into account the necessity of keeping the number of scans of the overall data as low as possible. The overall approach is summarized as follows:

Build the local models: At each local site, a hierarchical structure of representative points is built. A randomized approach to the selection of the representative points has been found to be successful in terms of the overall number of scans of the data required. For each scan of the data, a set of new representative points are found. These are assigned to the existing representative points (for any scan after the first), and all of the unassigned non-representative points are assigned to a representative point. If a point is within a radius determined by the user defined allowable error, it is considered to be properly positioned, and is not reassigned. Otherwise, it is marked as needing further refinement, and will be repositioned during subsequent scans of the data. It should be noted that if a point is out of radius, but is assigned to a representative point (note that all points will be assigned to a representative point after the first scan), that point will be assigned to one of the child representative points of the current representative point for the point in question.

Build the global model: Once the local models are generated in the above described manner, they are sent to the central facilitator site. Here, the points are assembled into a global model. This is accomplished by building a hierarchical structure by combining the local models, and using the representative points to express the features of each of the individual points. The distance between any two points is defined as the distance between its representative points. As the number of representative points is much less than the overall number of points in the data set, this is far more efficient than simply transmitting the overall data to the central site.

Examine areas of interest: If interesting areas within the data are found to be worthy of further examination, full feature information can be requested from the sites for the points within that area of the feature space for closer examination.

The following section presents the Collective approach for Bayesian learning.

6.4. Collective Bayesian Learning

In this section, we discuss the Collective approach to learning a Bayesian network (BN) that is designed for a distributed data scenario. This involves learning a local model from the individual databases, identifying a subset of the observations at each site that is most relevant for discovering patterns across databases, transmitting these observations to a central site and discovering relationships among variables across sites, and forming an aggregate model based on the local models and the discovered relationships across sites.

Bayesian Networks. A BN is a probabilistic graph model. It can be defined as a pair $(\mathcal{G}, P)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed acyclic graph (DAG) (Jen96; Hec98). Here, $\mathcal{V}$ is the vertex set which represents variables in the problem, $\mathcal{E}$ is the edge set which denotes probabilistic relationships among the variables, and P is a probability function defined on the set of vertices $\mathcal{V}$. Figure 6 is an example of a BN called the ASIA model (adapted from (LS88)). Based on the conditional independence implied by the DAG, the joint distribution of all variables is:

$$\begin{aligned} P(A,S,T,L,B,E,X,D) = {} & P(A)P(S)P(T \mid A)P(L \mid S) \\ & P(B \mid S)P(E \mid T,L)P(X \mid E)P(D \mid B,E). \end{aligned}$$

Two important issues in using a BN are : (a) learning a BN and (b) probabilistic inference. Learning a BN involves learning the structure of the network (the directed graph), and obtaining the conditional probabilities (parameters) associated with the network. Once a BN is constructed, we usually need to determine various probabilities of interest from the model. This process is referred to as probabilistic inference.

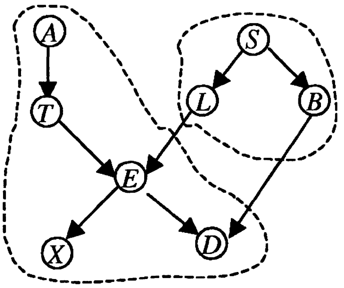


Figure 6. ASIA Model

Collective Bayesian Network Learning Strategy. We now present a collective strategy to learn the BN from data in distributed sites. Learning a BN for the homogeneous case was studied by (Ken97). We address here the heterogenous case.

The primary steps in our approach are: (a) Compute local BNs (local model) involving the variables observed at each site based on local data, (b) At each site, based on the local BN, identify the observations that are most likely to be evidence of coupling between local and non-local variables and transmit a subset of these observations to a central site, (c)

At the central site, a limited number of observations of all the variables are now available. Using this, compute a non-local BN consisting of links between variables across two or more sites, (d) Combine the local models with the links discovered at the central site to obtain a collective BN. The non-local BN thus constructed would be effective in identifying associations between variables across sites, whereas the local BNs would detect associations among local variables at each site. A collective BN can be obtained by taking the union of nodes and edges of the local BNs and the nonlocal BN and using the conditional probabilities from the appropriate BNs.

We now describe an approach to selecting observations at the local sites that are most likely to be evidence of strong coupling between variables at two different sites. The key idea of our approach is that the samples that do not fit well with the local models are likely to be evidence of coupling between local and non-local variables.[4] We transmit these samples to a central site and use them to learn a collective BN.

Selection of samples for transmission to global site. For simplicity, we will assume that the data is distributed between two sites and will illustrate the approach using the BN in Figure 6. The extension of this approach to more than two sites is straightforward. Let us denote by A and B, the sites consisting of variables in the left and right groups, respectively, in Figure 6. At each local site, a local BN can be learned using only samples in this site. Let $l_A(.)$ and $l_B(.)$ denote the likelihood functions for the local model obtained at sites A and B. The observations at each site are ranked based on how well it fits the local model, using the local likelihood functions. Let $S(A)$ denote the set of keys associated with observations having low likelihood under $l_A(.)$; i.e. we set a threshold ρ_A and if $l_A(x) \leq \rho_A$, then observation $x \in S(A)$. The observations corresponding to the set of keys in $S = S(A) \cap S(B)$ are then transmitted to the central site.

In a sense, our approach to learning the cross terms in the BN involves a type of *importance sampling*, where we select the observations that are most likely to have high conditional probabilities corresponding to the terms involving variables from both sites. Naturally, when the values of the variables (features) from the different sites, corresponding to these selected observations are pooled together at the central site, we can learn the coupling links as well as estimate the associated conditional distributions.

Experimental Results. Our experiments were performed on a dataset that was generated from the BN depicted in Figure 6. Note that there are two edges ($L \to E$ and $B \to D$) that connect variables from site B to site A, the rest of the six edges being local.

First, we tested the ability of the collective approach to detect the two non-local edges. Local BNs were constructed using a conditional independence test based algorithm (CBL97), for learning the BN structure and a maximum likelihood based method for estimating the conditional probabilities. The local BNs were exact as far as the edges involving only the local variables. The observations that had low likelihood at both sites were then sent to the central site. It was observed that, by transmitting just 5% of the total data, an error-free non-local BN was obtained at the central site.

Next, we assessed the accuracy of the estimated conditional probabilities. The Kullback-Leibler (KL) distance or relative entropy between the conditional probabilities was computed based on our collective BN and a BN obtained using a centralized approach (by transmitting all data to one site), for the terms that involve variables from both the sites; viz. $p(E \mid T, L)$ and $p(D \mid E, B)$. Given that these are conditional probabilities, we compute the sum over all the possible values of $\{T, L\}$, of the KL distance between $p_{\text{coll}}(E \mid T, L)$ and $p_{\text{cntr}}(E \mid T, L)$, estimated using our collective approach and the centralized approach, respectively. Figure 7 depicts the KL distance for nodes E and D. Clearly, even with a small data communication, the estimates of the conditional probabilities of the cross-terms, based on our collective approach, is quite close to that obtained by the centralized approach.

Our preliminary experiments clearly demonstrates the feasibility of our collective learning strategy. However, a number of theoretical and practical issues must be resolved before a collective BN learning strategy can become reality. In particular, we need to address the following issues: (a) Theoretical properties of our probability estimators like consistency, bias, variance, (b) Scalability of the approach with respect to number of sites, and number of variables, and (c) Optimal tradeoff choices between communication overhead and performance.

7. Conclusions

This chapter presents a collection of different DDM techniques that can be used for developing a comprehensive suite to mine distributed and heterogeneous data. The chapter focussed on heterogeneous data since there exists a relatively larger body of work on homogeneous DDM. In this chapter we considered the collective data mining algorithms in the context of a simple abstraction of heterogeneity that depends on the features observed at different sites. This abstraction can be easily

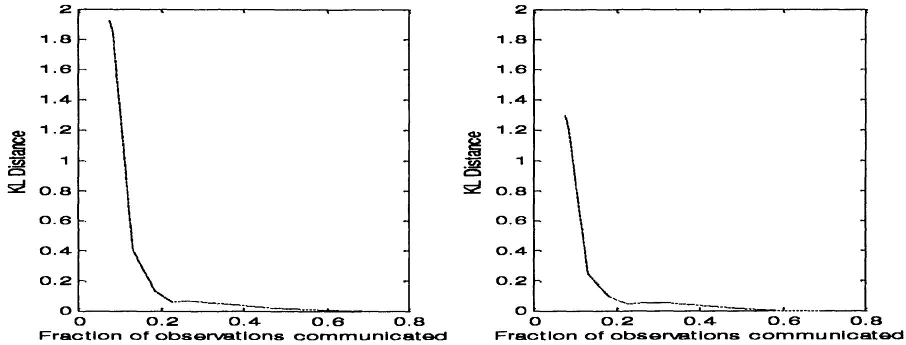


Figure 7. KL Distance between the conditional probabilities computed using the collective and centralized approaches for node E (left) and node D (right).

specialized for various situations like applications with multiple tables per site, multiple associations between events observed at different sites, overlapping features among different sites, and others. Although at this point of time we have a reasonable collection of DDM algorithms developed by different research groups, there are many issues that need more attention. DDM system architecture, mobile DDM, distributed visualization and other human-computer interaction issues are some of them. We strongly believe that once these issues are explored the DDM technology will significantly enhance the capabilities of the state-of-the-art scientific data mining systems for large distributed and mobile environments.

Acknowledgments

This research is supported by the United States National Science Foundation CAREER Award IIS-0093353.

Notes

1. `http://eospso.gsfc.nasa.gov`
2. `http://kdd.ics.uci.edu/databases/pioneer/pioneer.html`
3. For a non-binary string, $f(x) = \sum_j w_j \overline{\psi}_j(x)$ where $\psi_j(x)$ is a complex Fourier basis function and $\overline{\psi}_j(x)$ is its conjugate.
4. We assume here that the data is noise-free.

References

[AKPB97] J. Aronis, V. Kulluri, F. Provost, and B. Buchanan. The WoRLD: Knowledge discovery and multiple distributed databases. In *Proceedings of the Florida Artificial Intelle-*

gence Research Symposium (FLAIRS-97), pages 11–14, 1997. Also available as Technical Report ISL-96-6, Intelligent Systems Laborotory, Department of Computer Science, University of Pittsburgh.

[BS97] R. Bhatnagar and S. Srinivasan. Pattern discovery in distributed databases. In *Proceedings of the AAAI-97 Conference*, pages 503–508, Providence, 1997. AAAI Press.

[CBL97] J. Cheng, D. A. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. In *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*, 1997.

[Col87] J.R. Coloquhoun. A decision tree method of forecasting thunderstorms, severe thunderstorms and tornadoes thunderstorms, severe thunderstorms and tornadoes. *Weather and Forecasting*, 2:337–345, 1987.

[CS99] V. Crestana and N. Soparkar. Mining decentralized data repositories. Technical Report CSE-TR-385-99, University of Michigan, Ann Arbor, MI, 1999.

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[FS97] Y. Freund and R. E. Schapire. A decision theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):97–119, August 1997.

[GRS01] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. *SIGMOD Rec. (USA)*, 27(2):73–84, 2001.

[Hec98] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models.* Kluwer Academic Publishers, 1998.

[HK99] D. Hershberger and H. Kargupta. Distributed multivariate regression using wavelet-based collective data mining. Technical Report EECS-99-02, School of EECS, Washington State University, 1999. To be published in the Special Issue on Parallel and Distributed Data Mining of the Journal of Parallel Distributed Computing, Guest Eds: Vipin Kumar, Sanjay Ranka, and Vineet Singh.

[Jac91] J. E. Jackson. *A User's Guide to Principal Components.* John Wiley, 1991.

[Jen96] F. Jensen. *An Introduction to Bayesian Networks.* Springer, 1996.

[JK99] E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In *Lecture Notes in Computer Science*, volume 1759, pages 221–244. Springer-Verlag, 1999.

[Ken97] Y. Kenji. Distributed cooperative Bayesian learning strategies. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pages 250–262, Nashville, Tennessee, 1997. ACM Press.

[KHK+00] H. Kargupta, W. Huang, S. Krishnamrthy, H. Park, and S. Wang. Collective principal component analysis from distributed, heterogeneous data. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the Principles of Data Mining and Knowledge Discovery Conference*, volume 1910, pages 452–457, Berlin, September 2000. Springer. Lecture Notes in Computer Science.

[KHKJ00] H. Kargupta, W. Huang, S. Krishnamurthy, and E. Johnson. Distributed clustering using collective principle component analysis. In *Proceedings of the ACM SIGKDD Workshop on Distributed and Parallel Knowledge Discovery in Databases*, pages 8–19, August 2000.

[KN96] R. King and M. Novak. Supporting information infrastructure for distributed, heterogeneous knowledge discovery. In *Proceedings of SIGMOD 96 Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, Canada, 1996. `http://www.cs.colorado.edu/~sanctuary/Papers/datamining.ps`.

[KP01] H. Kargupta and B. Park. Decision trees tickers for mining data streams. In communication, 2001.

[KPHJ00] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 133–184. AAAI/ MIT Press, Menlo Park, California, USA, 2000.

[LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[LPO00] A. Lazarevic, D. Pokrajac, and Z. Obradovic. Distributed clustering and local regression for knowledge discovery in multiple spatial databases. In *Proc. 8th European Symposium on Artificial Neural Networks*, Bruges, Belgium, April 2000.

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, series B*, 50:157–224, 1988.

[MSG00] S. McClean, B. Scotney, and K. Greer. Clustering heterogeneous distributed databases. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000*, pages 20–29, Boston, 2000.

[PAK01] B Park, R. Ayyagari, and H. Kargupta. A fourier analysis-based approach to learn classifiers from distributed heterogeneous data. Chicago, IL, USA, 2001. To be published in the Proceedings of the 1st SIAM International Conference on Data Mining.

[PB95] F. J. Provost and B. Buchanan. Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20:35–61, 1995.

[PKA01] B. Park, H. Kargupta, and R. Ayyagari. Distributed predictive mining, decision trees, and fourier analysis. Technical Report, in preperation., 2001.

[PKJ⁺00] H. Park, H. Kargupta, E. Johnson, R. E. Sanseverino, L. D. Silvestre, and D. Hershberger. Distributed, collaborative data analysis from heterogeneous sites using a scalable evolutionary technique. Technical Report EECS98-001, School of Electrical Engineering and Computer Science, Washington State University, 2000.

[PO00] S. Parthasarathy and M. Ogihara. Clustering distributed homogeneous datasets. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the Principles of Data Mining and Knowledge Discovery Conference*, pages 566–574, September 2000.

[PS00] A. Prodromidis and S. Stolfo. Cost complexity-based pruning of ensemble classifiers. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000*, pages 30–40, Boston, 2000.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[SCD⁺97] S. Salzberg, X. Chen, A. Delcher, J. Henderson, and K. Fasman. MORGAN: A decision tree system for finding genes in

DNA. Technical Report JHU-97/02, Dept. of Computer Science, Johns Hopkins University, Baltimore, MD 21218, 1997.

[SCF+95] S. Salzberg, R. Chandar, H. Ford, S. K. Murthy, and R. White. Decision trees for automated identification of cosmic ray hits in hubble space telescope images. *Publications of the Astronomical Society of the Pacific*, 107:1–10, May 1995.

[Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Sib73] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16:30–34, 1973.

[SS00] M. Sayal and P. Scheuermann. A distributed clustering algorithm for web-based access patterns. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000*, pages 41–48, Boston, 2000.

[TG00a] K. Tumer and J. Ghosh. Robust order statistics based ensemble for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery*, pages 185–210. MIT, 2000.

[TG00b] A. Turinsky and R. Grossman. A framework for finding distributed data mining strategies that are intermediate between centeralized strategies and in-place strategies. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000*, pages 1–7, Boston, 2000.

Chapter 17

DECOMPOSABLE ALGORITHMS FOR DATA MINING

Raj Bhatnagar

Abstract Most data mining algorithms have been designed and developed with the assumption that all the relevant data is resident at a single node of a computer network. In a networked environment the data relevant for a mining task may reside, in components, on a number of geographically distributed computer nodes. These databases cannot be easily moved to other network sites due to their size, ownership and security considerations. Mining algorithms, therefore, are needed that can decompose themselves to match the nature of data distribution across nodes and execute partial computations at local data sites and then compose these local summaries to construct global results. We present some mining algorithms having such dynamic decomposition capability.

Keywords: Decomposable algorithms, mining distributed databases

1. Introduction

Geographic distribution of data across a network and the need for various data-components at processors executing a mining algorithms can be viewed as belonging to one of the following scenarios. *Locally Resident Data:* Many scientific simulations generate terabytes of data that typically reside in storage systems at a single computer site. An *intensive computing* system is needed to process such large locally resident data [HKK00, AS96]. *Locally Needed Remote Data:* Some applications, such as weather simulations, require satellite image data that is available from a remote site. In these situations all the image data must be transferred and made available at the processing site. *Remotely Consulted Data:* Some mining applications require that a number of large databases be consulted for completing the mining tasks but the actual database tuples are not needed to be physically transferred to any particular site.

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 307–317.
© 2001 *Kluwer Academic Publishers.*

A mobile Java *aglet*, traveling from one participating network site to the other can carry partial computations of a mining algorithm with it, and complete the algorithm as it moves and accesses the databases at successive sites. However, it requires a redesign of the mining algorithms so as to minimize the number of hops for the aglet, or for the exchange of messages among nodes. These algorithms would significantly reduce the size of the data that may have to be transferred among nodes for a mining task.

Examples: The first example is from the domain of Geographic Information Systems. A database D_1 of county-to-county historical migrations for US is available at one network site. The attributes of this database are: (i) County Id #1, (ii) County Id #2, (iii) Migration-1-to-2 (year-1), (iv) Migration-2-to-1 (year-1); and attributes (iii) and (iv) are repeated for each of the 20+ years for which data is available. There are 3000+ counties in the US, and therefore, this database is likely to have approximately one million records. Another network site contains the census database, from which one can extract a table D_2 for counties, containing 260+ attributes for each county. It is desired to mine the knowledge contained jointly in D_1 and D_2. The preferred alternative is to leave the databases at their native sites and execute a decomposable algorithm that knows what summaries to obtain from each site at each step of the algorithm and combine these summaries to obtain the global results. Domains of Genomics and Engineering Design are other examples where knowledge is geographically distributed and it is difficult to integrated it at a single node.

We present here an insight into algorithms that can decompose their computations to fit the nature of data distribution across the participating database nodes, while also minimizing the number of messages to be exchanged among the nodes.

2. Decomposable Algorithms

Let us say we have n databases located at n different network nodes and they together constitute a dataset $\mathcal{D}$ for some data mining task. We model databases D_is, at the i^{th} sites, by a *relation* containing a number of tuples. The set of attributes contained in D_i is represented by X_i. For a pair of relations D_i and D_j the sets X_i and X_j may have a set of shared attributes given by S_{ij}. Since an arbitrary number of independent databases may be included in any particular mining task, we cannot assume any kind of normalization to have been done with their schemas. The implicit and global dataset $\mathcal{D}$ with which the mining task is to be performed consists of a subset of the set of tuples generated by a *Join* operation performed on all D_is.

However, the tuples of $\mathcal{D}$ cannot be made explicit at any one network site because entire databases, D_i's, cannot be moved to other sites. The tuples of $\mathcal{D}$, therefore, must remain only implicitly specified. This inability to make explicit the tuples of $\mathcal{D}$ is the main problem addressed by the decomposable algorithms. To facilitate computations with implicitly specified $\mathcal{D}$ we define a set S that is the union of all the attribute intersection sets S_{ij}'s.

$$S = \cup_{i,j\ ,\, i \neq j} \quad S_{ij} \tag{17.1}$$

That is, S contains the names of all those attributes that occur in more than one D_i. We then define a relation *Shareds* which contains all possible tuples that can be enumerated for the attributes in the set S.

Each participating database D_i is represented by an agent C_i that communicates with similar agents at other nodes to exchange simple statistical summaries. Also, in case of an implementation using mobile *aglets*, it is the agents C_is that act as an interface between the database and a visiting *aglet*. The decomposable data mining algorithms can be seen to reside with these agents and each agent is capable of initiating and completing a data mining task by either exchanging summaries with other agents (stationary agents), or by visiting other sites (mobile agents).

Let us say a result $\mathcal{R}$ is obtained by applying a function $\mathcal{F}$ to a dataset $\mathcal{D}$. That is, R = F(D). When we are inducing a decision tree from $\mathcal{D}$ the value of $\mathcal{R}$ is the induced decision tree and when we are finding association rules, $\mathcal{R}$ is a set of selected candidate item sets. Here $\mathcal{F}$ corresponds to an algorithm for computing $\mathcal{R}$ from $\mathcal{D}$.

The dataset $\mathcal{D}$ cannot be made explicit and is known only implicitly in terms of its explicit components $D_1, D_2, \ldots, D_n$. The implementation of $\mathcal{F}$ above can be redesigned by a functionally equivalent formulation:

$$\mathcal{R}(\mathcal{S}) = \mathcal{G}[g_1(D_1, S), g_2(D_2, S), \ldots g_n(D_n, S)]. \tag{17.2}$$

That is, a local computation $g_i(D_i, S)$ is performed at *Node-i* using the database D_i and the knowledge about the attributes shared among all the nodes. The results of these local computations are aggregated using the operation $\mathcal{G}$. The number and nature and g_i operators and the nature of $\mathcal{G}$ would vary with the D_is and the set of shared attributes S. Hence, a different set of g-operators would need to be generated for each new instance of D_is and S. The components operators of a decomposition (G and g_is), therefore, need to be dynamically determined for each instance of $\mathcal{F}(\mathcal{D})$ depending on the participating nodes; the attributes contained in their native databases; and the overlap pattern of their attributes as reflected in S.

3. Related Research

Extensive research exists relating to algorithms for sequential and parallel processor architectures [AHU74, Akl97, HKK00, Jaj92]. Main focus of these algorithms has been on systems of closely coupled processors where data can be easily shared by processors by mechanisms similar to that of Distributed Shared Memory. Algorithms for geographically distributed databases where data cannot be shared as easily as accessing a shared memory, have received little attention. In the shared memory formulation the primitives fetch some data from the memory of a different processor, but in a networked databases context the primitive visits the other node with a partial computation, its own control, and a few statements to be executed at the visited site before a summary is returned. Multi-agent systems research has addressed many issues relating to the distribution of knowledge and processing capability over a loosely connected communication network. In most of this work [DC87, HSK97, Sen96], agents are modeled as having only a limited view of the global knowledge. In contrast, our decomposable algorithms are directed at situations where cooperative agents can freely access each others' knowledge, visit each other's site to advance some of their partial computations, and thus learn concepts from their collective knowledge while minimizing the exchange of data among the nodes.

The work in distributed data mining [CS96, SFL$^+$97] seeks to learn local models completely at local sites and then resolve their differences at the coordinating learning site. Algorithms for computing association rules [AIS93, AS94, HKK00] address the cases when the database are huge but resides on a single site. Decomposable algorithms adapt these same basic algorithms with the added capability to decompose computations to suit the data distribution. Intelligent Query Answering and Data Clustering in large databases have been addressed in [HHCF96], and their treatment is also limited to databases residing at a single computer site.

4. Decomposing Arithmetic Primitives

When the tuples of a complete dataset $\mathcal{D}$ are explicitly available in a relation then the count of all its tuples can be obtained easily. For the case of implicitly defined $\mathcal{D}$, we can decompose the counting process in such a way that various *local* count requests can be sent to the sites of individual D_is and the responses can then be composed to reconstruct the *total* count for the tuples in implicit $\mathcal{D}$. A simple algebraic decomposition for obtaining the count N_{total} is as follows:

$$N_{total} = \sum_{J}(\prod_{t=1}^{n}(N(D_t)_{cond_J})) \tag{17.3}$$

where the subscript $cond_J$ specifies a condition composed from the attribute value pairs of the J^{th} tuple of the relation *Shareds*, n is the number of participating databases (D_is), and $(N(D_t)_{cond})$ is the count in relation D_t of those tuples that satisfy the condition $cond_J$. In comparison with the expressions in 17.2 we can see that in this decomposition

$$g_i(D_i, S) = N(D_t)_{cond_J} \tag{17.4}$$

where J corresponds to the J^{th} tuple of *Shareds*. One such summary is needed from each database site for each tuple of the relation *Shareds*. The function $\mathcal{G}$ in this case would perform a sum-of-products from the summaries as per the equation 17.3 above. Each term in the product is the count of tuples satisfying condition $cond_J$ in a D_i. The resulting product produces the number of distinct tuples that would be contributed to the implicit *Join* of all the D_is for the condition specified by $cond_J$. The summations in the above expression amount to selecting each tuple of *Shareds* as $cond_J$ and then summing the product terms obtained for each tuple. This expression, therefore, simulates the effect of a *Join* operation performed on all the n databases without explicitly enumerating the tuples. A desirable aspect of the above decomposition of N_{total} is that each local computation,(g_i, $(N(D_t)_{cond})$ can be translated into an SQL query: *Select count (*) where* $cond_J$ and sent to the site containing database D_t.

Complexity: When there are k attributes in the relation *Shareds* and each attribute has l distinct possible values on an average, the relation *Shareds* would have $k * l$ tuples in all. If there are n database sites then we would be sending one request to each site for each tuple in *Shareds*, amounting to a total of $n * k * l$ messages being exchanged. However, it is possible to send a request to an i^{th} site for all $g_i(D_i, S)$ values, that is, values corresponding to all tuples $cond_J$ of S in one request and receive the summaries in one message. This is the step that takes this implementation from a *low granularity* decomposition to a *higher granularity* decomposition. This reduces the number of messages exchanged to n, the same as the number of participating nodes. Also, if all the n requests are sent out at the same time and execute simultaneously at the participating nodes, then the time taken for entire decomposed computation is the same as that taken by the longest elapsed time for an exchange between the two nodes.

The *Average* value for an attribute B in $\mathcal{D}$ can be computed in a decomposable manner as follows:

$$Avg_B = (\frac{1}{N_{total}}) * \sum_{j=1}^{l}(B_j * N(B_j)) \tag{17.5}$$

where N_{total} is the total number of tuples in $\mathcal{D}$, $B_1 \ldots B_l$ are the possible discrete numeric values of attribute B, and $N(B_j)$ is the count of tuples in the dataset in which B takes the value B_j.

For each of the l different values of attribute B we would need to exchange n messages and another n messages would need to be exchanged to determine N_{Total}. The average value can, therefore, be computed by exchanging $(l + 1) * n$ messages among the participating nodes.

5. Decomposing Association Rules Algorithms

Algorithms for discovering *Association Rules* in databases [AIS93, AS94] can be easily redesigned in their decomposable versions. The main phases of these algorithms involve iterating the following two steps: (i) Enumerate item-sets at level k from the frequent item-sets determined at level $k - 1$; and (ii) Determine the *support* and the *confidence* levels for the item-sets and rules at level k. A generalizable decomposition of the association rule algorithm can be implemented as follows. An agent at any one of the network nodes initiates and controls the algorithm. This agent performs the task of maintaining and generating the candidate item-sets. Computation of *support* and *confidence* levels requires consultation of the database $\mathcal{D}$ and it is this step that needs to be redesigned for decomposition.

Support for candidate sets: The decomposition for obtaining the *count* of tuples described above can be adapted to count only those tuples that satisfy certain condition ($new - condition$) by simply changing $Cond_J$ in equation 17.3 above to

$$N_{new-condition} = \sum_{J}(\prod_{t=1}^{n}(N(D_t)_{cond_J.and.new-condition})) \tag{17.6}$$

This adaptation would be required to determine the support level for a candidate set. The values of attributes in a candidate set would form the *new-condition.* This resulting candidate count divided by the count N_{total} provides the support level for a candidate set of attribute-value pairs. Each count would require n messages to be exchanged and thus a support level can be computed by exchanging $2 * n$ messages. However, count requests for all candidate sets under consideration in an iteration of the algorithm can be packed into a single message. Therefore, the support levels for all the candidate sets in an iteration can be computed by exchanging only $2 * n$ messages among the nodes. If an association rule algorithm needs to run up to k levels (iteration) then we need to exchange a total of $2 * n * k$ messages among the nodes to completely run the association rules algorithm. This number of messages is not dependent on the number of tuples contained in each database and the algorithm, therefore, is easily scalable to large databases. Also, this number of messages is much smaller than the data that may need to be transferred if we were to accumulate all databases at one site and then perform the data mining task.

6. Induction of Decision Trees

We can also easily decompose [BS97] a decision tree induction algorithm that uses minimization of average entropy as a tree-building heuristic. It is because entropy computation depends only on various counts obtained from the databases. Various tree induction algorithms [BFOS84, Qui86], start by considering the complete dataset $\mathcal{D}$ belonging at the root of the tree and then repeating the following steps until all or a large majority of tuples at each leaf node of the tree belong to some unique class. (i) Pick one such dataset at a leaf node, some large fraction of whose tuples belong to different classes; (ii) Select an attribute a_j having m distinct values: $a_{j1}, a_{j2} \ldots a_{jm}$. The attribute that results in minimum average entropy for the resulting partitions is chosen. This entropy value can be computed by the decomposition primitives described above, mainly the counts with various conditions placed on tuples for determining the appropriate probability values. (iii) Split D into m distinct partitions such that the k^{th} partition contains only those tuples for which $a_j = a_{jk}$. (iv) The m distinct partitions are added to the tree as child datasets of the partitioned parent dataset. These child nodes reside at the end of m branches emanating from the parent node.

In the preceding discussion we have included the complexity of performing decomposition of each computational step in terms of the number of messages to be exchanged among the nodes. We show below an expression for the number of messages that need to be exchanged, by transferring only one $g_i(D_i, S)$ summary at a time, for generating a simple decision tree using the above algorithm. Let us say: (i) There are n databases, $D_1 \ldots D_n$, residing at n different network sites. (ii) There are k attributes in set S of shared attributes. Each attribute in this set appears at more than one site. (iii) There are m distinct attributes in

$\mathcal{D}$ ($\cup_{i=1}^{n} A_i$) combined. (iv) There are l possible discrete values for each attribute in set S.

The informational entropy that is computed by the algorithm at each leaf node of the growing decision tree is given by the expression:

$$E = \sum_{b=1}^{m} (\frac{N_b}{N_t} \times (\sum_{c} - \frac{N_{bc}}{N_b} log_2 \frac{N_{bc}}{N_b}), \tag{17.7}$$

where N_b is the total number of tuples in the database at a parent node of the tree, N_b is the number of tuples at a child node, and N_{bc} is the number of tuples in N_b that belong to class c. To evaluate the entropy for a database once, we need to compute the following quantities: (i) one N_t count; (ii) l N_b counts; and (iii) l^2 N_{bc} counts. The computation of each entropy value, therefore, requires an exchange of $n * (l^{k+2})$ messages among the participating nodes. For a dataset at depth d in the decision tree the number of messages exchanged would be $(m - d) * n * (l^{k+2})$. If we assume that on the average the decision tree is akin to a filled $l - ary$ tree with p levels then the total number of message-exchanges needed would be: $(n * l^{k+2}) \sum_{d=0}^{p} (m - d) * l^p$

The above expression gives an estimate of the number of messages to be exchanged for constructing a decision tree using the decomposed version of algorithm, assuming that only one integer value is exchanged per message. However, using a more efficient implementation in which all the needed $g_i(D_i, S)$ values are requested from a node in a single message, we need only $c * n$ messages to compute a count or a sum of products where c is a constant and n is the number of participating nodes (As described earlier for the case of counts). In this case an entropy computation would require an exchange of only $l * n$ messages and computation of $d*l$ entropy values would require $d*l*l*n$ messages to be exchanged.

7. Decomposing Covariance Matrix Computation

Principal components and eigen values and vectors are used extensively in many data mining algorithms. The basic computational task we need to perform for evaluating all the above is the determination of the covariance matrix for the implicit dataset $\mathcal{D}$. All the arithmetic primitives encountered till now were first order quantities and computation of covariance matrix requires us to compute second order quantities, such as the expected values of the sums of products of pairs of variables. For such *sum of products* we need to compute $\sum_i x_i y_i$ where x and y are two attributes and i is an index over all tuples in $\mathcal{D}$. Due to the different ways in which attributes x and y may be distributed across the participating databases, it turns out that there are six different cases, each requiring a different way to decompose the computation of the sum-of-products. The details for decomposing each case are presented in [BY99].

The six cases are: (i) $\sum_i x_i^2$ *when* $x \in S$: , that is, x is a shared attribute. (ii) $\sum_i x_i^2$ *when* $x \notin S$: , that is, x is not a shared attribute and it resides on a single network node. (iii)$\sum_i x_i y_i$, that is, both $x, y \in Shareds$ (iv) $\sum_i x_i y_i$, that is, only one of the two attributes is in the shared set S. (v) $\sum_i x_i y_i$ when both x and y are not in the set of shared attributes and x and y reside on different network nodes. (vi) $\sum_i x_i y_i$ when $x, y \notin$ shared, and x, y reside on same node. These six cases cover all possible ways in which a pair of attributes may be distributed across the network and require a different way of decomposition. The node initiating the pattern discovery task knows the attributes residing at every other node and can initiate the decomposition computations according to the case encountered. With Sums-of-Products of attribute pairs we now have everything that is needed for computing the covariance matrix of an implicit $\mathcal{D}$, by using a set of arbitrarily overlapping projections of $\mathcal{D}$. The network node desiring to compute the covariance matrix for an implicit $\mathcal{D}$ can compute the various components for the following computation:

$$E\left[(X-\mu_x)(X-\mu_x)^T\right] = \begin{bmatrix} \rho_{11}^2 \cdots \rho_{1n}^2 \\ \vdots \\ \rho_{n1}^2 \cdots \rho_{nn}^2 \end{bmatrix}$$

where $\rho_{ij}^2 = E\left[(X_i-\mu_i)(X_j-\mu_j)\right] = E\left[X_i X_j\right] - \mu_i \mu_j)$. The covariance matrix can be constructed at any one of the participating nodes by building the sums of products, means and differences as they are required in the above expression.

8. Conclusion

We have demonstrated that counts of tuples, averages, candidate support levels, informational entropy values and covariance matrices can be computed in a distributed knowledge environment without moving all the databases to a single site. Also, for simple arithmetic computations, the number of messages to be exchanged among the n participating nodes for completing an arithmetic primitive does not exceed order of n.

We have also demonstrated the adaptability of association rule algorithms, an informational entropy driven decision tree induction algorithm, and also the computation of covariance matrices in distributed database situations.

One very significant contribution of these results is that many data mining algorithms can be executed on a number of databases residing at different network nodes without having to move them to a single site and the communication cost of such decomposable algorithms is also very low; much lower than the cost of bringing the databases to a single site and then executing the data mining algorithms.

Acknowledgments

The author acknowledges the support provided by NSF through their grant number IIS-0081434 which had supported part of the research described in this paper.

References

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison Wesley, Reading, MA, 1974.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD 93*, pages 207–216, 1993.

[Akl97] Selim J. Akl. *Parallel Computation: Models and Methods.* Prentice Hall, New Jersey, 1997.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[AS96] R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, December 1996.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[BS97] R. Bhatnagar and S. Srinivasan. Pattern discovery in distributed databases. In *Proc. of the AAAI 1997*, pages 503–508, 1997.

[BY99] R. Bhatnagar and B. Young. Computations in distributed knowledge environments. In *Proceedings of ADCOM-99*, pages 83–89, Roorkee, India, 1999.

[CS96] Philip K. Chan and Salvatore J. Stolfo. Sharing learned models among remote database partitions by local meta-learning. In *Proceedings of KDD 1996*, pages 2–7, 1996.

[DC87] E. H. Durfee and V. R. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11), 1987.

[HHCF96] J. Han, Y. Huang, N. Cercone, and Y. Fu. Intelligent query answering by knowledge discovery techniques. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):373–390, 1996.

[HKK00] Eui-Hong Han, George Karypis, and Vipin Kunar. Scalable parallel data mining for association rules. *IEEE Transactions on Knowledge and Data Engineering*, 12(3), May/June 2000.

[HSK97] Michael N. Huhns, Muninder P. Singh, and Tomasz Ksiezyk. *Readings in Agents.* Morgan Kaufmann Publishers, 1997.

[Jaj92] Joseph Jaja. *An Introduction to Parallel Algorithms.* Addison Wesley, Reading, MA, 1992.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Sen96] S. Sen. Adaption co-evolution and learning in multi-agent systems. In *Proceedings of 1996 AAAI Symposium on Multi-AGent Learning*, 1996.

[SFL+97] S. J. Stolfo, D. Fan, W. Lee, A. Prodromidis, and P. Chan. Jam: Java agents for meta-learning: Issues and initial results. In *Proceedings of AAAI97 Workshop on AI Methods in Fraud and Risk Management*, pages 2–7, 1997.

Chapter 18

HDDI™: HIERARCHICAL DISTRIBUTED DYNAMIC INDEXING

William M. Pottenger, Yong-Bin Kim, and Daryl D. Meling

Abstract The explosive growth of digital repositories of information has been enabled by recent developments in communication and information technologies. The global Internet/World Wide Web exemplifies the rapid deployment of such technologies. Despite significant accomplishments in internetworking, however, scalable indexing and data-mining techniques for computational knowledge management lag behind the rapid growth of distributed collections. Hierarchical Distributed Dynamic Indexing (HDDI™) is an approach that dynamically creates a hierarchical index from distributed document collections. At each node of the hierarchical index, a knowledge base is created and subtopic regions of semantic locality in conceptual space are identified. This chapter introduces HDDI™, focusing on the model building techniques employed at each node of the hierarchy. A novel approach to information clustering based on the contextual transitivity of similarity between terms is introduced. We conclude with several example applications of HDDI™ in the textual data mining and information retrieval fields.

Keywords: Textual data mining, information retrieval, machine learning, computational knowledge management, artificial intelligence, HDDI™

1. Introduction

Current developments in computer technology are radically changing the nature of distributed information management. Clearly, widespread digitization of information and the ubiquity of networking have created fundamentally new possibilities for collecting and distributing information. Just as clearly, it is critical that new information infrastructure be developed that enables effective mining of the huge volume of distributed information emerging in digital form [PCP00]. A recent article estimated that no more than about 16% of the publicly accessible web has been indexed by any single search engine [LG99].

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 319–333.
© 2001 *Kluwer Academic Publishers.*

Traditional methods of indexing combine multiple subject areas into a single, monolithic index. There are enough documents on the Web that such indexing technology often fails to perform effective search. The difficulty lies in the fact that since so many documents and subjects are being combined together, retrieving all the documents that match a particular word phrase often returns too many documents for effective use. This problem has been known for some time [NRC92]. Solutions based on link analysis have mitigated these difficulties yet fail to adequately address issues of recall [BP98].

In order to properly address this problem, a paradigm shift is needed in the approach to indexing. First and foremost, it is clear that digital collections are now and will continue to be distributed. Our first premise is thus that *indices* must also be distributed. Secondly, it must be realized that the information contained in these distributed digital repositories can be classified in a hierarchical manner. Traditionally, knowledge hierarchies, or ontologies, have been created with human expertise. One popular form is the thesaurus (e.g., the National Library of Medicine's MeSH thesaurus). Such an approach does not scale to the tremendous amount of emerging digital information for two reasons: as knowledge increases, new topics are emerging at a greater rate, and both this and the sheer volume of information preclude manual approaches to indexing. Our second premise is thus that distributed indices must properly reflect the hierarchical nature of knowledge. Thirdly, due to the vast increase in communications bandwidth and computing and online storage capabilities mentioned above, digital collections are frequently updated. This process reflects a key characteristic of 21st century collections: namely, they are dynamic in nature. Our third premise is thus that any new information infrastructure must include *dynamic* indexing capabilities.

In the final analysis, these three technologies must be integrated into a cohesive whole: HDDI™, *Hierarchical Distributed Dynamic Indexing.* HDDI™ is a novel approach to organizing large quantities of unstructured data in a loosely coupled distributed environment under development at Lehigh University and at the National Center for Supercomputing Applications. The approach is based on the algorithmic creation of subtopic regions of semantic locality in sets of distributed documents; this allows automatic discovery of similarities at a fine level of granularity amongst concepts within documents. In this way, hierarchical indices (such as those created now "by hand" in many places on the web; www.yahoo.com is probably the most well-known example) are generated for topics in documents in a volatile, distributed environment, providing the information seeker with an always up-to-date map of information spaces. The ability to generate large hierarchical indices on the fly allows for a realistic, useful mapping of cyberspace without the need for time-consuming human intervention. This technique is most valuable when applied to items within some institutional zone — to map out, for instance, large sets of corporate or scientific documents. Here, subjective issues relating to "importance" or "quality" can be sidestepped, and the power of the HDDI™ strategy can be fully leveraged—an unstructured set of documents lacking any sort of metadata can be bound to a hierarchical knowledge structure generated automatically based on word frequencies.

HDDI™ is a natural outgrowth of advances in technology that leverage existing, inherited knowledge structures popular in today's web-based networked environment. As HDDI™ technology develops, we are discovering novel approaches that address several issues of managing distributed digital information within the context of the HDDI™ paradigm.

This chapter will focus on the core computational knowledge management algorithms employed in building models at the nodes of hierarchical indices. Although the focus of this book is on data mining of scientific data sets, we do not limit our discussion to scientific data in this chapter, but rather consider textual data in all forms. This is only natural given the ubiquitous employment of the written word in human endeavor.

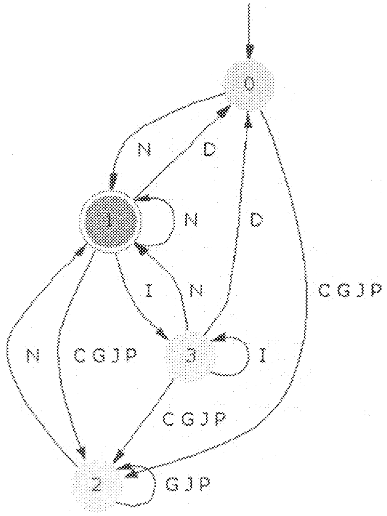
In the following sections we present an outline of HDDI™ technology employed in building hierarchical indices. We close with a summary of applications in which HDDI™ technology is being deployed.

2. Building a Hierarchical Index

The following steps are involved in the process of building a hierarchical index: concept identification/extraction; concept co-occurrence matrix formation; hierarchy construction; knowledge base creation; identification of regions of semantic locality; and hierarchy mapping. Each of these steps is outlined in more detail below. Several aspects of this approach reflect our initial intuition on how the problem should be addressed. Each of these six steps is being addressed in the course of our research in order to refine our approach.

2.1. Concept Identification/Extraction

Our approach to concept identification/extraction includes the following three steps: input item (document) parsing, part of speech tagging and concept identification. The parsing stage takes SGML, HTML or generalized XML tagged items as input. We have utilities to convert from a variety of input formats to XML, including proprietary airline safety data and US government patent data. Based on AI techniques [C00], [B92], our part of speech tagging approach includes the use of both lexical and contextual rules for identifying various parts of speech. After identifying each word's part of speech, we invoke a finite-state machine (pictured below) that accepts maximal length English-language noun phrases. [BCGKMP01], [K96].



A Finite State Automaton for Recognizing Complex Noun Phrases State 0 is the start state, and state 1 the final state. C is a cardinal number, G a verb (gerund or present participle), P a verb (past participle), J an adjective, N a noun, I a preposition and D a determiner.

Our enhanced state-machine identifies and extracts concepts consisting of complex noun phrases composed of multiple modifiers, including gerund verb forms. The final result of these three steps is a reformulation of the original collection that includes a summation of the location and number of occurrences of each extracted concept. The next stage of the process receives this reformulated collection.

2.2. Concept Co-occurrence Matrix Formation

"Co-occurring" defines concepts that occur within the same item. An item can be defined as an intelligently created logical unit of text that is cohesive semantically. Examples include abstracts, titles, web pages, airline safety incident reports, patents, etc. The co-occurrence relation is reflexive and symmetric but not transitive. Given concepts extracted by the above process, we compute concept frequency and co-occurrence matrices. We also compute the frequencies of co-occurrences of concept pairs among all items in the set.

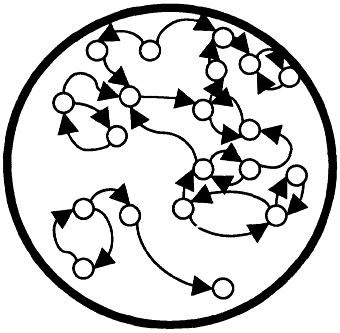
The literature discusses various definitions of co-occurrence [BYRN99]. Our approach incorporates measures based on proximity as well as techniques that dynamically define the extent of sub-items within a given item (see Section 2.5). Our preliminary results indicate that this latter approach is crucial in dealing with the full text of items. We also reported on research in parallelizing the computation of such semantic relations based on the theory of *coalescing loop operators* [P98]. Similar techniques are being applied to scale the computational information management algorithms developed as part of this research to large collections.

2.3. Hierarchy Construction

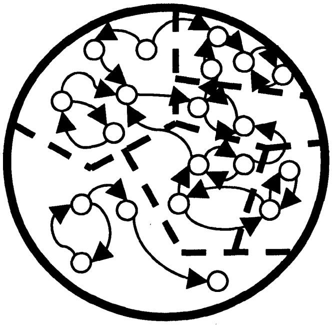
Hierarchy construction is a meta-level organizational process that combines the matrices formed in the previous step. These co-occurrence matrices provide the basis for organizing concepts into an ontology of knowledge based on the content of the collections. Systematic filtering, pruning, and meshing lower level (child) matrices form the hierarchical structure, producing higher level (parent) combined matrices. The process is iterative in that matrices are pruned and merged at successively higher levels. The resulting hierarchical index consists of high-resolution leaf-level index nodes that become increasingly less precise (i.e., more general) as the hierarchy is built. This process is visualized in Section 3 below, and is a topic of ongoing research [K01].

2.4. Knowledge Base Creation

Knowledge base creation is the second meta-level organizational process. For each matrix in the hierarchy constructed in the previous step, and for each concept in each matrix we compute a similarity with other concepts. This one-to-many mapping associates each concept with a list of related concepts ranked by similarity. Co-occurring concepts are ranked in decreasing order of similarity. More general concepts occur lower in the list. Each concept pair is weighted, creating asymmetric measures of pair-wise similarity between



Knowledge Base



Knowledge Neighborhood

concepts. The similarity is a mapping from one concept to another that quantitatively determines how similar they are semantically. We term the resultant mapping a *knowledge base*[1]. A knowledge base is represented as an asymmetric directed graph in which nodes are concepts and arc weights are similarity measures. The knowledge base can be visualized as a graph, illustrated by example on the left below, in which vertices represent concepts and edges represent the pair-wise similarity between concepts.

In [P97] techniques were implemented that produce a knowledge base using an extension of the statistical model developed in [CL92]. The model building techniques are based on a *cluster function* defined as follows [CMNS97]:

$$ClusterWeight\left(C_j, C_k\right) = \frac{\sum_{i=1}^{n} d_{ijk}}{\sum_{i=1}^{n} d_{ij}} \times WeightingFactor\left(C_k\right)$$

$$ClusterWeight\left(C_k, C_j\right) = \frac{\sum_{i=1}^{n} d_{ikj}}{\sum_{i=1}^{n} d_{ik}} \times WeightingFactor\left(C_j\right)$$

These two equations indicate the cluster weights, or similarity, from concept C_j to concept C_k (the first equation) and from concept C_k to concept C_j (the second equation). d_{ij} and d_{ik} are the product of concept frequency and inverse document frequency and are defined in a similar manner (this is a variation of the popular *tf*idf* measure used primarily in text-based vector space modeling where *tf* is concept frequency and *idf* is inverse document frequency [S89]). d_{ij}, for example, is defined as:

$$d_{ij} = tf_{ij} \times \log\left(\frac{N}{df_j} \times w_j\right)$$

where N represents the total number of documents in the collection, tf_{ij} is the frequency of occurrence of concept C_j in document i, df_j is the number of documents (across the entire collection of N documents) in which concept C_j occurs, and w_j is the number of words in concept C_j.

d_{ijk} and d_{ikj} represent the combined weights of both concepts C_j and C_k in document i and are also defined in a similar manner. d_{ijk}, for example, is defined as follows:

$$d_{ijk} = tf_{ijk} \times \log\left(\frac{N}{df_{jk}} \times w_j\right)$$

---

[1] Note that follow-on work that builds on [CL92] terms this a *Concept Space.*

Here tf_{ijk} represents the minimum number of occurrences of concept C_j and concept C_k in document i. df_{jk} represents the number of documents (in a collection of N documents) in which concepts C_j and C_k occur together. The final expression, w_j, is the number of words in concept C_j.

In order to penalize general concepts that appear in many places in the co-occurrence analysis, a weighting scheme similar to the inverse document frequency function is employed. C_j, for example, has the following weighting factor:

$$WeightingFactor(C_j) = \frac{\log \frac{N}{df_j}}{\log N}$$

Concepts with a higher value for df_j (i.e., more general concepts) have a smaller weighting factor, which results in a lower similarity. Co-occurring concepts are ranked in decreasing order of similarity, with the result that more general concepts occur lower in the list of co-occurring concepts.

Ongoing research includes enhancement of this cluster function to account for several additional factors including, for example, metrics such as the ratio of commonly used to total words in a concept.

Our high-performance implementation for computing knowledge bases employs cluster functions of this nature in a parallel C^{++} application optimized to achieve efficiencies of over 100% on the SGI/Cray Origin2000 and SGI Power Challenge supercomputers [P98]. Based on the theory of *coalescing loop operators*, the outermost loop in this application was parallelized [P97]. In conjunction with this outer-loop parallelism, a custom memory manager enabled the superlinear speedups reported in the table following on the SGI Power Challenge. Runtimes are recorded as wall-clock execution time for repositories of different sizes.

	Num. Procs.	CR h:m:s	CR S_p	DR h:m:s	DR S_p
Serial	1	1:17:19	1	10:30:27	1
Parallel	2	0:27:16	2.84	3:32:49	2.96
	4	0:14:28	5.34	1:53:52	5.54
	8	0:08:41	8.90	1:08:20	9.23
	16	0:05:44	13.49	0:39:22	16.01

Speedups on SGI Power Challenge

2.5. Identification of Regions of Semantic Locality

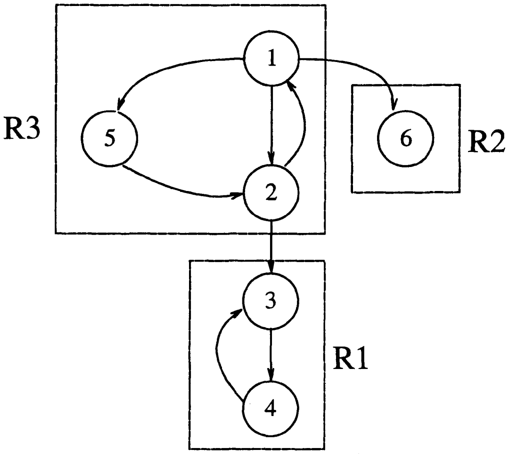
The resulting weight assignments from knowledge base creation are context-sensitive. We use these weights to determine regions of semantic locality (i.e., conceptual density) within each collection. We then detect clusters of concepts within a knowledge base [BP00], [B99], [T72].

The result is a knowledge base consisting of regions of high-density clusters of concepts – subtopic regions of semantic locality. These regions consist of clusters of concepts that commonly appear together and collectively create a *knowledge neighborhood.* Our premise is that we can impute a constrained, *contextual transitivity* to the co-occurrence relation [BP00], thereby forming regions of semantic locality (see illustration on the right in section 4 above). The motivation for the use of the term semantic locality comes from the commonly applied premise that grouping similar concepts together leads to increased effectiveness and efficiency in query search and retrieval [SJ71]. Note however that the similarity relation is by definition not transitive. The theoretical basis for our algorithm, sLoc, is a concept we term *contextual transitivity* in the similarity relation. In essence this means that a threshold is chosen based on the structure and distribution of the similarities in the knowledge base and transitivity is constrained accordingly.

Contextual transitivity extends Schütze's conceptualization of *second order co-occurrence* [S98] by using *n-order* co-occurrence, where *n* varies with the underlying semantic structure of the model. Here is a simple example of how sLoc employs contextual transitivity to group concepts together in a region of semantic locality. Let us consider the three concepts *Java*, *applet* and *e-commerce*. If the concepts occur in the same document, they are called co-occurring concepts. Let us assume that *Java* and *applet* co-occur and that *Java* and *e-commerce* also co-occur. This means that there is at least one document that contains both *Java* and *applet*, and that there is at least one document that contains both *Java* and *e-commerce*, but *applet* and *e-commerce* do not necessarily co-occur. It may, however, be useful to gather the three concepts in one unique class and infer that *e-commerce* and *applet* are actually related. That is what we mean by contextual transitivity. Note that similarity is not a boolean relation – rather, it can take on a range of values. Two concepts can thus be more or less similar. The characteristics of the structure of a given knowledge base lead to a heuristic minimum similarity value needed for two concepts to occur in the same cluster, and therefore to *constrain* transitivity. It is called *contextual* because this minimum similarity value depends on the structure and distribution of the similarities in the knowledge base.

The computational core of sLoc is based on an algorithm due to Tarjan [T72]. Tarjan's algorithm uses a variant of depth-first search to determine the strongly connected components of a directed graph. This was the first algorithm to solve the problem in linear time. This is an important feature due to the fact that graph clustering is an NP-hard problem and the only heuristics we are aware of are not linear. The theory can be found in [AHU74]. The figure below depicts the operation of Tarjan's algorithm as it identifies strongly connected regions (R_1, R_2, R_3) in a simple graph.

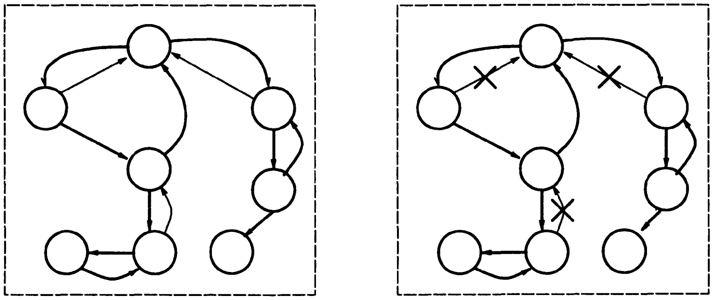
Before tackling the algorithm itself we must first introduce the following notation: Let $\mathcal{N}$ be the set of nodes i in the input graph, and let N be the total number of nodes. Let $\mathcal{A}$ be the set of arcs in the input graph and A the total number of arcs. An arc $a_{ij} \in \mathcal{A}$ connects node i to node j. Let $\mathcal{W}$ be the set of arc weights in the graph and $w_{i,j}$ the weight of the arc going from node i to node j. Therefore $\mathcal{W} = \{w_{i,j}\}_{(i,j) \in \mathcal{N}^2}$.



A knowledge base is an asymmetric graph and therefore $w_{i,j}$ may differ from $w_{j,i}$. Moreover, if $a_{i,j} \notin \mathcal{A}$ then $w_{i,j} = 0$; in particular, for all i, $w_{i,i} = 0$. Finally, let M be the mean of the arc weights $M = 1/A \sum_{(i,j) \in N^2} w_{i,j}$ and SD the standard deviation of the distribution of arc weights $SD = \sqrt{1/(A-1) \sum_{(i,j) \in N^2} (w_{i,j} - M)^2}$.

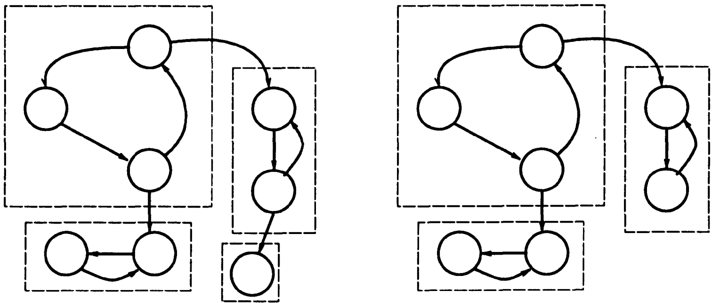
2.5.1. The sLoc Algorithm Step by Step

The figure below depicts the three steps of the sLoc process. Prior to the first step, the weights in the knowledge base are normalized (step 0 below). The first step in sLoc is to statistically prune the input graph. Arcs of weight smaller than a certain threshold τ are virtually pruned. Note that since the similarities are asymmetric, an arc from concept *a* to concept *b* can be pruned while the arc back from *b* to *a* remains. The second step involves the identification of the clusters within the graph. Tarjan's algorithm is applied to find strongly connected regions. At this stage each strongly connected region is a cluster. The size of a given cluster is the number of nodes (concepts) it contains. During the third and final step, clusters of size smaller than parameter *s* are discarded (they are assumed to be outliers). We interpret the remaining clusters as regions of semantic locality in the knowledge base. The greater τ, the more arcs are cut off, and therefore the smaller in size the strongly connected regions. Thus the greater τ the smaller in size and the more focused will be the regions of semantic locality.



(0) Standardize weights in the raw knowledge base

(1) Prune the graph

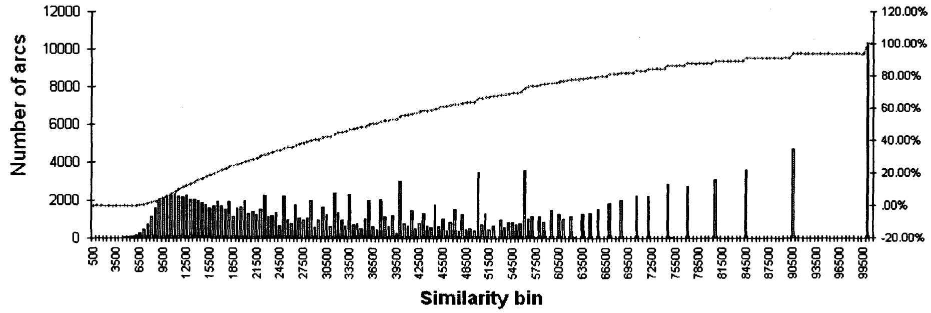


(2) Find strongly connected regions in the pruned graph

(3) Discard clusters of size smaller than s (here s = 2)

The question that has arisen is twofold: Given an input knowledge base, what threshold τ should be applied? Secondly, can we determine a value for the threshold that scales across domain and collection size?

Let us tackle the first question on how to determine the value of τ yielding an optimum clustering. Our premise is that the optimum τ can be determined statistically as a function of the mean *M*, the standard deviation *SD* and other knowledge base dependent variables (e.g., size, maximum weight, etc.). We have conducted preliminary experiments to study the distribution of weights in various knowledge bases. Our preliminary results indicate that the distribution of weights is quite consistent across both subject domain and collection size. The figure above represents one such common distribution.



This was computed from the MED gold standard information retrieval collection [S75].

It should be stressed that we have yet to compute a knowledge base that does not exhibit a distribution of this nature. Given this fact, we have developed the following heuristic for the threshold τ: $\tau(\alpha) = \mathbf{max}(w) - \alpha * \boldsymbol{SD}$. In this equation, τ is the cut-off weight used to prune the graph and α is the number of standard deviations. For example, $\tau(\mathbf{1/2})$ is the threshold corresponding to the maximum weight in the graph minus half of the standard deviation of the distribution of arc weights.

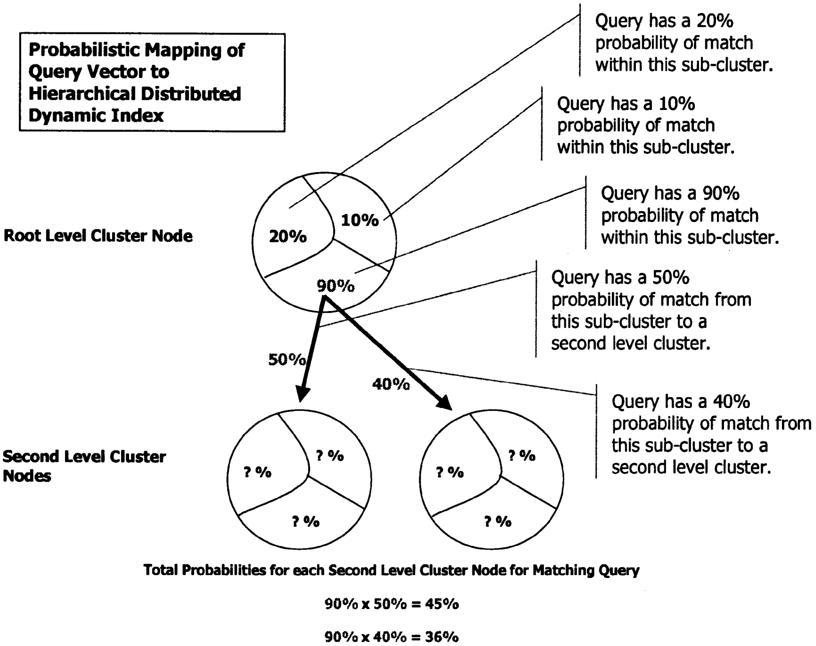
At this point in time we are still investigating answers to the second question posed above. We have conducted initial experiments in some of the application domains mentioned in Section 3 that indicate the range of α between [1.5, 2.0] is suitable [BP00], [Y00], [PY00].

2.6. Hierarchy mapping

The final process is to create a mapping of the information space by developing linkages between the clusters identified during semantic locality detection. These mappings supply the necessary paths between information areas at higher and lower levels of the hierarchy. Links between clusters are based on overlap of content between clusters as well as similarities in topology. This process defines relationships between clusters, so that clusters are interconnected both with the child clusters that they were merged from and the higher-level parent clusters they were used to create. Upon completion of this phase, each level of the hierarchical index contains sufficient information to determine its relative position as a knowledge area in the overall knowledge hierarchy. This is an area of ongoing research [K01].

3. Applications of HDDI™

One of the first applications of HDDI™ is in the organization of large distributed and dynamic data sets into a cohesive hierarchical framework for optimal probabilistic search. Our research objective is to create a distributed system whereby collections are automatically indexed hierarchically for use in concept-based knowledge searches as depicted in the figure below.


Probabilistic Mapping of Query Vector to Hierarchical Distributed Dynamic Index
Query has a 20% probability of match within this sub-cluster.
Query has a 10% probability of match within this sub-cluster.
Root Level Cluster Node
20%
10%
90%
Query has a 90% probability of match within this sub-cluster.
Query has a 50% probability of match from this sub-cluster to a second level cluster.
50%
40%
Query has a 40% probability of match from this sub-cluster to a second level cluster.
Second Level Cluster Nodes
? %
? %
? %
? %
? %
? %
Total Probabilities for each Second Level Cluster Node for Matching Query
90% x 50% = 45%
90% x 40% = 36%

Using the HDDI™ technology infrastructure, we are also investigating the automatic detection of emerging content. Trends in scientific research, technology forecasting, and automatic detection of emerging interpretations in case law are just a few examples of the variety of applications in which ongoing research that employs HDDI™ technology is being conducted [PY00].

We have also recently explored scaling HDDI™ technology from the textual to the numeric domain. HDDI™ techniques cluster *concepts* into classes. An example of a concept in textual data mining is a noun phrase. Our approach to asset management is to identify distinct *financial concepts* in financial data, and apply the HDDI™ model building and clustering-partitioning techniques to group financial concepts into regions exhibiting similar characteristics that can then be employed in predictive financial modeling.

Another area of HDDI™ research involves processing data that records both synchronous and asynchronous interactions between individuals. Using HDDI™ technologies, we are scaling our models to build relationship networks between people that can then be correlated with semantic regions of locality in the topics under discussion. Tools based on these methods provide insight into the types and range of topics and associations in social interactions and can be used, for example, to identify social and semantic links between groups of authors and collections of publications.

A related research initiative involves the design of a multimedia framework for constructive, inquiry-based learning for introductory and upper level computer science courses. The content is drawn from the field of Object-Oriented Software Engineering (OOSE). This National Science Foundation project includes development of HDDI™ textual data mining techniques for establishing temporal context that will be automatically employed by learners to detect emerging trends in OOSE research [BPK01]

4. Conclusion

We have outlined a set of core algorithms for building models in HDDI™, a framework for mining and management of distributed textual data. Current research includes the use of stochastic approximation to determine confidence intervals for optimal values of various parameters used in HDDI™ (such as the parameter α in sLoc), as well as the construction of an agent system that executes HDDI™ in a distributed environment. Ongoing work continues to focus on the validation of HDDI™ model building and indexing techniques in a variety of applications.

Acknowledgements

We gratefully acknowledge the assistance of our Lord and Savior Jesus Christ in completing this work. We also gratefully acknowledge the assistance of the many who worked together with us at the National Center for Supercomputing Applications and at Lehigh University to make this a reality.

References

[AHU74] V. Aho, J. E. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, MA.

[BCGKMP01] R. Bader, M. Callahan, D. Grim, J. Krause, N. Miller and William M. Pottenger, The Role of the HDDI^{TM} Collection Builder in Hierarchical Distributed Dynamic Indexing, *Proceedings of the Textmine '01 Workshop, First SIAM International Conference on Data Mining*, April.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto, Eds. Modern Information Retrieval, ACM Press, New York.

[BPK01] G. Blank, William M. Pottenger, G. D. Kessler, The CIMEL Project: Constructive, Collaborative, Inquiry-based Multimedia E-Learning, http://www.eecs.lehigh.edu/~cimel.

[B99] F. D. Bouskila, The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing and Information Retrieval, M.S. Thesis, Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, December (Bouskila's thesis work was supervised by William M. Pottenger).

[BP00] F. D. Bouskila and William M. Pottenger, The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing, *Proceedings of the International Conference on Artificial Intelligence (IC-AI'2000)*, Las Vegas, NV, June.

[B92] E. Brill, A Simple Rule-based Part of Speech Tagger, *Proceedings of the Third Conference on Applied Natural Language Processing*, ACL.

[BP98] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April.

[CL92] H. Chen and K. J. Lynch, Automatic Construction of Networks of Concepts Characterizing Document Databases, *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):885-902, September/October.

[CMNS97] H. Chen, J. Martinez, T. Ng and B. R. Schatz, A Concept Space Approach to Addressing the Vocabulary Problem in Scientific Information Retrieval: An Experiment on the Worm Community System, *Journal of the American Society for Information Science*, Volume 48, Number 1, January.

[C00] G. Cooke, SemanTag, gcooke@rt66.com.

[K96] L. Karttunen, Directed Replacement. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics,* Santa Cruz, California.

[K01] Y. B. Kim, The Role of Hierarchical Models in Hierarchical Distributed Dynamic Indexing, M.S. Thesis, Department of Computer Science at the University of Illinois at Urbana-Champaign, June.

[LG99] S. Lawrence and C. L. Giles, Accessibility of Information on the Web, Nature, Volume 400, pages 107—109.

[NRC92] National Research Council, *Computing the Future: A Broader Agenda for Computer Science and Engineering*, National Academy Press.

[P97] William Morton Pottenger, Theory, Techniques, and Experiments in Solving Recurrences in Computer Programs, Ph.D. thesis, Center for Supercomputing Research and Development in the Department of Computer Science at the University of Illinois at Urbana-Champaign, www.eecs.lehigh.edu/~billp/pubs/PhDThesis.ps, May.

[P98] William M. Pottenger, The Role of Associativity and Commutativity in the Detection and Transformation of Loop-Level Parallelism, In the *Proceedings of the 12th International Conference on Supercomputing*, www.eecs.lehigh.edu/~billp/pubs/2057.ps.gz, Melbourne, Australia, July.

[PY00] William M. Pottenger, Detecting Emerging Concepts in HDDI^{TM}. *Proceedings of the Computational Information Retrieval Workshop (CIR00)*, Raleigh, NC. October.

[PCP00] William M. Pottenger, M. R. Callahan, and M. D. Padgett, *Distributed Information Management,* Annual Review of Information Science and Technology (ARIST), The American Society for Information Science.

[S75] G. Salton, *Dynamic Information and Library Processing*, Prentice Hall, Englewood Cliffs, New Jersey.

[S89] G. Salton, *Automatic Text Processing,* Addison-Wesley Publishing Company, Inc., Reading, MA.

[S98] H. Schütze, Automatic Word Sense Discrimination, *Computational Linguistics,* vol. 24, no. 1, pp. 97-124.

[SJ71] K. Sparck-Jones, *Automatic Keyword Classification for Information Retrieval*, Butterworths, London, 1971.

[T72] R. E. Tarjan, Depth first search and linear graph algorithms, *SIAM J. Computing*, 1:146-160.

[Y00] T. Yang, Detecting Emerging Conceptual Contexts in Textual Collections, M.S. Thesis, Department of Computer Science at the University of Illinois at Urbana-Champaign, February.

Chapter 19

PARALLEL ALGORITHMS FOR CLUSTERING HIGH-DIMENSIONAL LARGE-SCALE DATASETS

Harsha Nagesh, Sanjay Goil, and Alok Choudhary

Abstract Clustering techniques for large scale and high dimensional data sets have found great interest in recent literature. Such data sets are found both in scientific and commercial applications. Clustering is the process of identifying dense regions in a sparse multi-dimensional data set. Several clustering techniques proposed earlier either lack in scalability to a very large set of dimensions or to a large data set. Many of them require key user inputs making it hard to be useful for real world data sets or fail to represent the generated clusters in a intuitive way. We have designed and implemented, pMAFIA, a density and grid based clustering algorithm wherein a multi-dimensional space is divided into finer grids and the dense regions found are merged together to identify the clusters. For large data sets with a large number of dimensions fine division of the multi-dimensional space leads to an enormous amount of computation. We have introduced an adaptive grid framework which not only reduces the computation vastly by forming grids based on the data distribution, but also improves the quality of clustering. Clustering algorithms also need to explore clusters in a subspace of the total data space. We have implemented a new bottom up algorithm which explores all possible subspaces to identify the embedded clusters. Further our framework requires no user input, making pMAFIA a completely unsupervised data mining algorithm. Finally, we have also introduced parallelism in the clustering process, which enables our data mining tool to scale up to massive data sets and large set of dimensions. Data parallelism coupled with task parallelism have shown to yield the best parallelization results on a diverse set of synthetic and real data sets.

Keywords: Data Mining, Clustering, Parallel Algorithms, Task and Data Parallelism, Massive data sets.

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 335–356.
© 2001 *Kluwer Academic Publishers.*

1. Introduction

Clustering has been extensively studied in statistics, machine learning [MS83], pattern recognition and image processing [Fuk90]. Intuitively, clustering techniques find *interesting* and previously unknown patterns in large scale data, embedded in a large multi-dimensional space and are applied to a wide variety of problems like customer segmentation [UF98], detection of clusters in geographic information systems [NH94a], etc. Clustering algorithms need to efficiently scale up with the dimensionality of data sets and also with the data base size. Noise present with data makes cluster detection harder. Further, clusters embedded in a subspace of the total data space, result in an explosion in the search space which is exponential in the data dimension. Parallelization of the clustering process adds the much needed computational power for these algorithms to be applied to massive financial data sets and large scale scientific data. Effective representation of the detected clusters is as important as cluster detection and improves its usability. Most of the earlier works in statistics and data mining [EKSX96, ZRL96] operate and find clusters in the whole data space. Most clustering algorithms [NH94a, APW^+99, ZRL96] require user input of several parameters like the number of clusters, average dimensionality of the cluster, etc., which are not only difficult to determine but are also not practical for real-world data sets. The output of these clustering algorithms are very sensitive to the input parameters.

We use a grid and density based approach for cluster detection in subspaces. Density based approaches regard clusters as higher density regions than their surroundings. The quality of results and the computation requirements heavily depend on the number of bins in each dimension. Larger bin sizes result in poor cluster quality, while finer bins result in an enormous amount of computation. Hence, determination of bin sizes automatically based on the data distribution greatly helps in finding correct clusters of high quality and reduces the computation substantially. However, subspace clustering algorithms need to explore all possible subspaces for embedded clusters in a bottom-up algorithm resulting in a time complexity which is exponential in the data dimension. Thus parallelization is the choice for grid based subspace clustering algorithms for massive data sets.

In this chapter we present pMAFIA (for Merging of Adaptive Finite Intervals), a scalable parallel subspace clustering algorithm using adaptive computation of the finite intervals (bins) in each dimension, which are merged to explore clusters in higher dimensions. The parallelization strategy involves both task and data parallelism with negligible communication overheads. Adaptive grid sizes improve the clustering quality by concentrating on the portions of the data space which have more points and thus are more likely to be part of a cluster region enabling minimal length DNF (disjunctive normal form) expressions, important for interpreting results by the end-user. Further, pMAFIA requires no user input, making it a completely unsupervised data mining algorithm. We describe recent work on clustering techniques in databases and density and grid based clustering in Section 2. Section 3 describes the parallel algorithms for detecting clusters and also explains the task and data parallel algorithms followed by their analysis. Section 4 presents the performance evaluation on a wide variety of synthetic and real data sets with large number of dimensions, highlighting both scalability of the algorithms and the quality of the clustering. Section 5 concludes the chapter.

2. Density and Grid Based Clustering

Clustering algorithms have long been studied in statistics and databases. k-means, k-mediods, CLARANS [NH94b], BIRCH [ZRL96], CURE [GRS98] are some of the earlier works. However, none of the above algorithms detect clusters in subspaces. PROCLUS [APW$^+$99], a subspace clustering algorithm finds representative cluster centers in an appropriate set of cluster dimensions. It needs the number of clusters, k, and the average cluster dimensionality, l, as input parameters, both of which are not known apriori for real data sets. Density and grid based approaches regard clusters as regions of data space in which objects are dense and are separated by regions of low object density (noise) [JD88]. The grid size determines the computations and the quality of the clustering. CLIQUE, a density and grid based approach for high dimensional data sets [AGGR98], detects clusters in the highest dimensional subspaces. It takes the size of the grid and a global density threshold for clusters as input parameters. A histogram is constructed by partitioning the data space into a number of non-overlapping regions and then mapping the data points to each cell in the grid. Equal length intervals are used in [AGGR98] to partition each dimension, which results in uniform volume cells. The number of points inside the cell with respect to the volume of the cell can be used to determine the density of the cell. The computational complexity and the quality of clustering is heavily dependent on grid size and density threshold parameters. A survey of parallel algorithms for hierarchical clustering using distance based metrics is given in [Ols95]. These are more theoretical PRAM algorithms. Recently, k-means algorithm has been parallelized [DM99], but is limited, however, in its applicability, as it requires the user to specify k, the number of clusters, and also does not find clusters in subspaces.

Clusters are unions of connected high density cells. Two k-dimensional cells are connected if they have a common face in the k-dimensional space or if they are connected by a common cell. Creating a histogram that counts the points contained in each unit is infeasible in high dimensional data. Subspace clustering further complicates the problem as it results in an explosion of such units. A bottom-up approach of finding dense units and merging them to find dense clusters in higher dimensional subspaces has been proposed in CLIQUE [AGGR98]. Each dimension is divided into a user specified number of intervals, ϵ. The algorithm starts by determining 1-dimensional dense units by making a pass over the data. In [AGGR98] candidate dense cells in any k dimensions are obtained by merging the dense cells in $(k-1)$ dimensions which share the *first* $(k-2)$ dimensions. However, this method of combining dense units does not explore all possible candidate dense cells in the framework of adaptive grids. For example, consider two 3-dimensional dense units $\{a_1, b_7, c_8\}$ and $\{b_7, c_8, d_9\}$, where (a, b, c, d) are the bins in the dimensions indicated by their subscripts. We can easily see that the two dense units results in a 4-dimensional candidate dense unit $\{a_1, b_7, c_8, d_9\}$ which is not formed by the approach in [AGGR98].

The algorithm presented in [AGGR98] works well when none of the dimensions are thrown away in the initial pass on the data set. However, in pMAFIA, the uninteresting dimensions in which data is uniformly distributed is identified in the initial pass over the data, which enables us to set a higher threshold for all the bins in such dimensions. In this process, the uninteresting dimensions get thrown away in the initial passes of the multi-pass subspace clustering algorithm. Thus, in our approach, candidate dense cells in k dimensions, are generated by merging any two dense cells, represented by an ordered set of $(k-1)$ dimensions, such that they share *any* of the $(k-2)$ dimensions. Although our algorithm seems to explore lot more candidate dense units than [AGGR98] due to the relaxed condition for combining two dense units, in reality it generates only the required candidate dense units as the adaptive grids formed enables us to capture the data distribution correctly in each dimension. However in [AGGR98] as all dimensions are treated equally one would end up with far more candidate dense units. In [AGGR98] candidate dense units are pruned based on a minimum description length technique to find the dense units only in *interesting* subspaces. However, as noted in [AGGR98] this could result in missing some dense units in the pruned subspaces. In order to maintain the high quality of clustering we do not use this pruning technique. Finally, a pass over data is made to find which of the candidate dense cells are actually dense. The algorithm terminates when no more candidate dense cells are generated.

2.1. Adaptive Grids

We propose an adaptive interval size in which bins are determined based on the data distribution in a particular dimension. The size of the bin and hence number of bins in each dimension in turn determine the computation and quality of clustering. Finer grids leads to an explosion in the number of candidate dense units, while coarser grids leads to fewer bins, and regions with noise data might also get propagated as dense cells. Also, a user defined uniform grid size may fail to detect many clusters or may yield very poor quality results. A single pass over the data is done in order to construct a histogram in every dimension. Algorithm Adaptive-Grid-Computation describes the steps of the adaptive grid technique. The domain of each dimension is divided into fine bins, such that each dimension has a minimum of 1000 fine bins. If the range of the dimension is from m to n then we set the number of fine bins in that dimension to be max(1000, $(n - m)$). The size of each such fine bin is different in each dimension and depends on m and n. A set of consecutive fine bins forms a window. The size of a window is determined by fixing the maximum number of adaptive grid bins that we would allow to be formed in a dimension. Adaptive grid bins are those formed by merging consecutive windows having similar histogram values, thereby forming variable sized adaptive grid bins. In our experiments we fix the maximum number of adaptive grid bins to be 20 per dimension. In essence, we are trying to find the best fit rectangular wave for the data distribution in a given dimension.

The maximum of the histogram value among the fine bins in a window is taken to reflect the window value. All the fine bins within the window are assigned the same maximum value. Adjacent windows (Adaptive grid bins) whose values differ by less than a percentage are merged together to form larger windows ensuring that we divide the dimensions into variable sized adaptive grid bins which capture the data distribution. However, in dimensions where data is uniformly distributed this results in a single adaptive grid bin and indicates much less likelihood of the presence of a cluster. In order to examine these dimensions further, we split the domain of such dimensions into a small fixed number of partitions (five in our runs) and collect statistics for these bins. This also allows us to set a high threshold as this dimension is less likely to be part of a cluster. This technique greatly reduces the computation time as we are able to limit the degree to which the bins from non-cluster dimensions contribute to the computation. In the dimensions with variable sized adaptive grid bins we set a variable threshold for each bin in that dimension. A bin in such a dimension is likely to be part of a cluster if it has a significantly (by a factor of α) greater number of points than it would have had, had the data been uniformly distributed in that dimension. Thus, for a bin of size a in a dimension of size D_i we set its threshold to be $\frac{\alpha a N}{D_i}$, where N is the total number of data points. A value of α greater than 1.5 has worked well in our experiments.

Adaptive Grid Computation

```
$D_i$ - Domain of $A_i$
$N$ - Total number of data points in the data set
$a$ - Size of a generic bin
for each dimension $A_i, i \in (1, \ldots d)$
    Divide $D_i$ into fine bins such that each dimension has atleast 1000 bins
    Compute the histogram for each fine bin of $A_i$, and set the value of
    the fine bins in a window to the maximum in the window
    From left to right merge two adjacent windows if they are within a threshold $\beta$
    /* Single bin implies an equi-distributed dimension */
    if(number of bins == 1)
        Divide the dimension $A_i$ into a fixed number of equal partitions.
Compute the threshold of each bin of size $a$ as $\frac{\alpha a N}{D_i}$
end
```

Figure 1(a) illustrates the uniform grid used in CLIQUE. This is not cognizant of the data distribution and generates many more candidate dense units for processing at each step than an adaptive grid illustrated in Figure 1(b). CLIQUE needs a post-processing phase to generate the minimal description length of the clusters to make the cluster definitions more amenable to the end-user. This is solved by using a greedy algorithm in CLIQUE which covers the found grids in clusters by maximal rectangles that provide coverage. Since this is an approximation of the cluster, it further adds to the complexity and reduces the correctness of the reported clusters.

On the other hand, since pMAFIA uses adaptive grid boundaries its cluster definitions are minimal DNF expressions and report the boundaries of clusters far more accurately. Figure 2 shows a cluster definition in two dimensions. The cluster reported by CLIQUE, *pqrs*, shown in Figure 2(a), loses the boundaries of the cluster. pMAFIA develops grid boundaries very close to the boundaries of the cluster and hence can incorporate most of the cluster in its definition. The DNF expression for the cluster *abcdefghijkl* shown in Figure 2(b) is $(l, y) \wedge (m, z) \wedge (n, y) \wedge (m, x) \wedge (m, y)$.

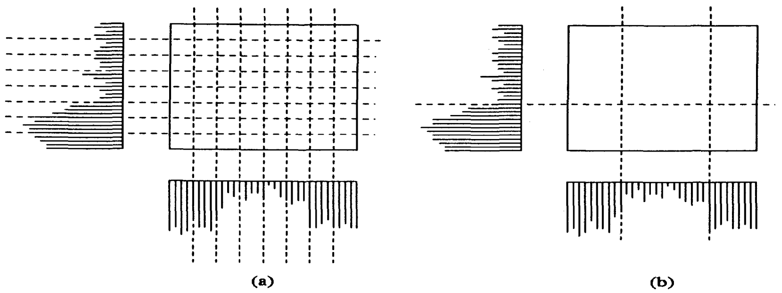


Figure 1. (a) Uniform grid size (b) Adaptive grid size

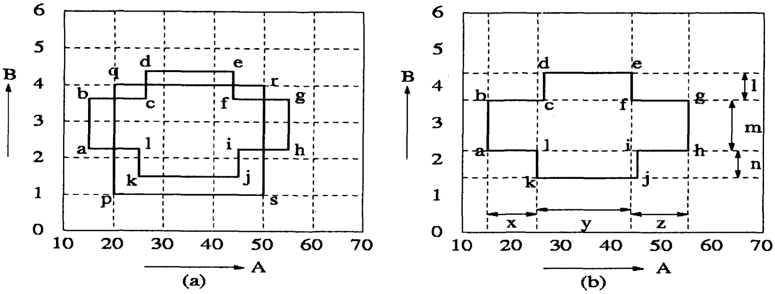


Figure 2. (a) Cluster discovered by CLIQUE (b) Cluster discovered by MAFIA

3. pMAFIA Implementation

We introduce a scalable parallel formulation of the subspace clustering algorithm incorporating both data and task parallelism on a distributed memory architecture. Programs run in the Single Program Multiple Data (SPMD) mode, where the same program runs on multiple processors but uses portions of the data assigned to the processor leading to data parallelism. Task parallelism is achieved by portions of the task at hand assigned to each processor. Processors communicate by exchanging messages. In contemporary parallel architectures which use this paradigm (e.g IBM SP2), the communication latencies are more than an order of magnitude larger than computation time. To achieve good parallelization, the overhead of communication has to be reduced by allocating tasks to processors such that relatively few small messages are exchanged. pMAFIA is a disk-based parallel and scalable algorithm which can handle massive data sets with a large number of dimensions. The algorithm can also run on a single processor in which the communication steps will be ignored. Algorithm pMAFIA shows the steps of the algorithm. In our set up on the IBM SP2, each processor reads a portion of the data from a shared disk initially and keeps it on the local disk. The bandwidth seen by a processor of an I/O access from the local disk is much higher than an access to a shared disk.

pMAFIA Algorithm

```
    $N$ - Number of records; $p$ - Number of processors; $d$ -Dimensionality of data
    $A_i$ - $i^{th}$ attribute $i \in d$; $B$ - Number of records that fit in memory buffer
allocated at each processor
    /* Each processor reads $\frac{N}{p}$ data from its local disk */
    On each processor
        Read $\frac{N}{pB}$ chunks of $B$ records from local disk and build a histogram in each
        dimension $A_i, i \in (1, \ldots, d)$
        **Reduce** communication to get the global histogram
        Determine adaptive intervals using the histogram in each dimension
        $A_i, i \in d$and also fix the threshold level
        Set candidate dense units to the bins found in each dimension
        Set current dimensionality, $k$ to 1
        while (no more dense units are found)
            if ($k > 1$)
                Find-candidate-dense-units();
            Read $\frac{N}{pB}$ chunks of $B$ records from local disk and for every record populate
            the candidate dense units
            **Reduce** communication to get the global candidate dense unit population
            Identify-dense-units();
            Register non dense units with the print data structures on the parent
            processor.
            Build-dense-unit-data-structures();
    if ( Parent Processor )
        print-clusters();
    end
```

pMAFIA consists mainly of the following steps. Candidate dense units in dimension k are built by combining dense units of dimension $k-1$ such that they share *any* of the $k-2$ dimensions. The parallel algorithm *Find-candidate-dense-units()* elaborates the steps involved in this process. The algorithm spends most of its time in making a pass over the data and finding out the dense units among the candidate dense units formed in every dimension. Repeated passes over the data need to be done as the algorithm progresses building dense units of higher dimensions. After finding the histogram count of the candidate dense units in a particular dimension, dense units are identified and dense unit data structures are built for the next higher dimension. The algorithms *Identify-dense-units()* and *Build-dense-unit-data-structures()* explain these in detail. We have introduced both data parallelism for the I/O intensive phase of building the histogram count of the candidate dense units and task parallelism for all the remaining tasks in the algorithm. The algorithm terminates when no more dense units exists. Clusters are finally printed by the parent processor at the end of the program.

3.1. Data Parallelism

Each processor starts by building a histogram for all dimensions with the data of size $\frac{N}{p}$, where N is the total number of records and p the number of processors. It is during this process that the data is read from the shared disk and written to the local disks of an IBM SP2 so that subsequent data set accesses can see a much larger bandwidth. A *Reduce* communication primitive with sum as its operand gathers the global histogram on each processor. Given a vector of size m on each processor and a binary associative operation, the *Reduce* operation computes a resultant vector of size m and stores it on every processor. The i^{th} element of the resultant vector is the result of combining the i^{th} element of the vectors stored on all the processors using the binary associative operation. Each processor now determines the adaptive finite intervals for every dimension and fixes the bin sizes and thresholds for every bin formed as elaborated in Adaptive Grid Algorithm. Each bin thus found is considered to be a candidate dense unit. Candidate dense units are populated by a pass on local data, which is read in chunks of B records. This enables our algorithm to be applicable to out of core data sets. Since each processor accesses only the local data, of size $\frac{N}{p}$, data parallelism can be obtained. A *Reduce* operation gets the global histogram count of the candidate dense units on all processors. This is followed by the task parallel algorithms which identify the dense units and build their data structures.

3.2. Task Parallelism

The task of finding the candidate dense units and identifying the dense units among the candidate dense units is divided among the processors such that each processor gets an equal amount of work. Optimal task partitioning is important in order to ensure that processors do not wait for other processors to finish their tasks. After completion of the job at hand, processors exchange messages in order to obtain the global information. The benefits of task parallelism are obtained only when the computation time is much more than the communication overheads. Each candidate dense unit (CDU) and, similarly a dense unit, in the d^{th}-dimension is completely specified by the d dimensions of the unit and their corresponding d bin indices. In our implementation we store this information in the form of an array of bytes, one array for the bin indices of all the CDUs and one for the CDU dimensions. Similarly an array each is used to store the dimensions and the bin indices of the dense units. By storing the information in the form of a linear array of bytes we not only optimize for space, but also gain enormously while

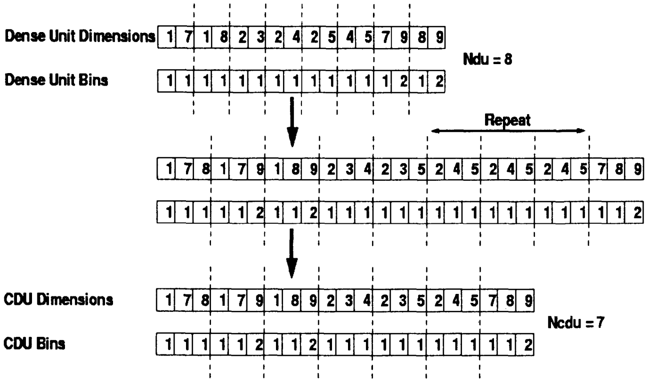

Dense Unit Dimensions
1 7 1 8 2 3 2 4 2 5 4 5 7 9 8 9
Dense Unit Bins
1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2
Ndu = 8
Repeat
1 7 8 1 7 9 1 8 9 2 3 4 2 3 5 2 4 5 2 4 5 2 4 5 7 8 9
1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
CDU Dimensions
1 7 8 1 7 9 1 8 9 2 3 4 2 3 5 2 4 5 7 8 9
CDU Bins
1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2
Ncdu = 7

Figure 3. Building CDUs. (Dimension, k = 3)

communicating. This helps in communicating information in a single step with the use of much smaller message buffers.

Building Candidate Dense Units. Figure 3 illustrates the process of building candidate dense units in dimension 3 for a data set in 10 dimensions. Candidate dense units in any dimension k are formed by combining dense units of dimension $k-1$ such that they share *any* $k-2$ dimensions. The steps of this algorithm is shown in Find-Candidate-Dense-Units Algorithm. Let the number of dense units be denoted by Ndu and the number of CDUs by $Ncdu$. It is easy to see that each dense unit needs to be examined with every other dense unit to form candidate dense units and this would lead to a huge amount of computation when Ndu is large. One can thus generate these CDUs in parallel and speedup the whole process. We shall now derive formulations which achieve *optimal* task partitioning of the candidate dense unit generation process and result in good speedups. Let k be the dimension in which we build candidate dense units. If Ndu is the number of dense units, it can be easily seen that the total amount of computation performed is $\frac{Ndu(Ndu+1)}{2}$ when each processor works on all the dense units. Let $n_1, n_2, \ldots, n_{p-1}$ be real numbers such that $0 < n_1 < n_2 < \ldots < n_{p-1} < Ndu$ and divide the Ndu dense units into p parts such that each part of the dense unit array be processed by one of the p processors. The division of Ndu into p parts is such that each processor does an equal amount of work, equal to $\frac{Ndu(Ndu+1)}{2p}$. If processor of rank i operates in the region between n_i and n_{i+1} comparing the dense units from n_i to n_{i+1} with all the other dense units greater than n_i, we have

$$Ndu * (n_{i+1} - n_i) - \left(\sum_{j=n_i}^{n_{i+1}-1} j \right) = \frac{Ndu(Ndu+1)}{2p} \qquad (19.1)$$

Solving the $p-1$ equations iteratively for $n_1, \ldots, n_{p-1}$, starting from n_1, one can obtain an optimal task partition for finding the candidate dense units. Having obtained the solution for n_i, the value of n_{i+1} can be obtained by solving the above quadratic equation. CDUs generated by the processors are communicated to the parent processor which concatenates the CDU dimension and bin arrays in the rank order of the processors. This information is *broadcast* to all the processors. Dense units which could not be combined with any other dense units are registered with the parent processor as a potential cluster in dimension $k-1$. Candidate dense units are generated in parallel only when each processor is guaranteed to have a minimal amount of work. If $Ncdu$ is less than a constant τ, CDUs are generated by all processors by processing all the dense units.

It can be seen from Figure 3 that the process of CDU generation may lead to identical candidate dense units being formed. We need to identify the repeated CDUs and retain only the unique elements. Elimination of identical CDUs is not only necessary but also greatly reduces the time during the task parallel part of the algorithm. One needs to compare each CDU with every other CDU to identify the repeated elements, which results in an $O(Ncdu^2)$ complexity algorithm. Thus, when $Ncdu$ is large we identify the repeated candidate dense units in parallel. This is similar to the generation of the candidate dense units and task parallelism is obtained by solving the above $(p-1)$ equations iteratively and with Ndu being substituted by $Ncdu$. Further, if the number of unique candidate dense units identified is still large, $(> \tau)$, we construct the data structures of the CDUs in parallel and finally exchange messages to obtain the global information. The steps of the algorithm are elaborated in Eliminate-repeat-CDUs() Algorithm.

Identification of Dense Units. Candidate dense units have to be examined to determine which of them are actually dense. Each processor selects $\frac{1}{p}^{th}$ of the candidate dense units to determine the dense units. The histogram count of each CDU is compared against the threshold of all the bins which form the CDU. A CDU is identified as dense if this histogram count is greater than the threshold of all the bins which form the CDU. A local count of the number of dense units found is maintained on each processor. A *Reduce* communication operation is now performed so that all processors have the information of the dense units from the set of candidate dense units. Another *Reduce* communication operation is performed to obtain the global count of the number of dense units (Ndu) on all processors. If the number of candidate dense units is less than τ, each processor works on all the candidate dense units

Find-candidate-dense-units()

```
Ncdu - Number of candidate dense units; CDU - Candidate dense unit
Ndu - Number of dense units; p - Number of Processors; τ - A constant
On each Processor
    if ( Ndu > τ )
        Find the start and end indices of the portion of the dense unit array it has
        to process.
        Build CDUs from its portion of the dense units.
        Send the information of local CDU count, CDU dimensions and Bins to the
        parent processor.
        Send non combinable dense unit information to the parent processor.
        if ( Parent Processor )
            Recv local CDU count, CDU dimension, CDU bin indices from all
            processors.
            Compute total number of CDUs, Ncdu and concatenate the CDU
            dimension
            and CDU bin indices in their rank order.
            Broadcast Ncdu and the concatenated CDU dimension and bin
            information.
            Recv non combinable dense unit information and update the data
            structures used for printing clusters.
        Eliminate-repeat-CDUs();
    else
        Build candidate dense units of dimension k from all the dense units of
        dimension (k − 1).
        if ( Parent Processor )
            Register non combinable dense units with the data structures used for
            printing clusters.
        Eliminate-repeat-CDUs();
    end
```

to determine the dense units. Algorithm Identify-dense-units() shows the steps involved. As before, if the number of dense units is greater than τ, the data structures of the dense units are constructed in parallel. It is important to note that we need to carefully divide the task of building the data structures as the dense units would not be distributed evenly. A linear search over the dense unit array is required to determine the start and end indices between which each processor operates for equal task distribution. Data structures of dense unit bin indices and their dimensions constructed in parallel are now merged with a *Reduce* communication operation. The steps of the algorithm is shown in Build-dense-unit-data-structures() Algorithm. The algorithm then proceeds to a higher dimension and starts building the candidate dense units. It terminates when there are no more candidate dense units.

After the cluster detection process is completed, the parent processor processes all the entries registered in its data structures for printing clusters. Clusters which are a proper subset of a higher dimension cluster are eliminated and only unique clusters of the highest dimensionality are presented to the end user. This increases the usability of the algorithm to a much greater extent.

```
Eliminate-repeat-CDUs()
    Nrepeat - repeated CDUs;
      On each Processor
       if ( Ncdu > $\tau$ )
          Find the start and end indices of the portion of the CDU
          array it has to process.
          Identify repeated CDUs in the entire CDU array as
          compared to the CDUs of its portion of the array.
          Reduce communication to obtain the global information
          of the repeated CDUs.
          Set the value of Nrepeat and $Ncdu = Ncdu - Nrepeat$.
          build-cdu-with-unique-elements();
       else
          Identify the repeated CDUs. Set the value of Nrepeat
          and $Ncdu = Ncdu - Nrepeat$.
          build-cdu-with-unique-elements();
    end

    build-cdu-with-unique-elements(){
      if ( Ncdu > $\tau$ )
          Divide Ncdu by p and find the start and end indices of
          the CDU array it has to process.
          Build the $(\frac{1}{p})^{th}$ CDU dimension and Bin arrays
          removing the repeated CDU elements.
          Send the $(\frac{1}{p})^{th}$ CDU information formed
          to the parent processor.
          if ( Parent Processor )
              Recv the $(\frac{1}{p})^{th}$ CDU information from
              all the processors. Concatenate them in their rank order.
              Broadcast the global CDU information to all processors.
          else
             Build CDU data structures of the CDU dimensions and bins
             removing the repeated CDU elements.
    }
```

3.3. Analysis

Let k represent the highest dimensionality of any dense unit in the data set. The running time of the algorithm is exponential in k. This is due to the fact that if a dense cell exists in k dimensions, then all its projections in a subset of k dimensions, $O(2^k)$ combinations, are also dense. Thus one needs to check for clusters in all possible subspaces among the k cluster dimensions. However, with the use of adaptive grids the number of candidate dense units is greatly reduced and thus enables pMAFIA to scale gracefully with the dimensionality of the data set and the data set size. Let α be the constant for communication and S be the size of messages exchanged among processors and N, the total number of records. Also, let B be the number of records that fit in memory buffer on each processor and let γ be the I/O access time for a block

```
Identify-dense-units()
On each Processor
  if ( $Ncdu > \tau$ )
    Divide $Ncdu$ by $p$. Find the start and end indices of the
    portion of the CDU array which it has to process.
    For each CDU in its portion of the array, compare the CDU
    histogram count with the thresholds of the bins which form the CDU.
    Determine if the CDU is dense or not.
    Maintain a local count of the number of dense units detected.
    Reduce communication to get the dense unit information
    of all the CDUs followed by another Reduce
    communication to get the total number of dense units $Ndu$.
  else
    For all CDUs, $Ncdu$, compare the CDU histogram count with
    the thresholds of the CDU bins.
    Determine if the CDU is dense or not. Find the total
    number of dense units $Ndu$.
end
```

```
Build-dense-unit-data-structures()
On each Processor
  if ( $Ndu > \tau$ )
    Divide $Ndu$ by $p$. Find the start and end indices of the
    CDU array on which it has to process.
    Construct the data structures related to the dimension and
    bin indices of the dense units from its portion of the
    CDU array.
    Reduce communication to get the global information of
    the data structures of the dense units.
  else
    From all CDUs, construct data structures related to
    the dimension and bin indices of the dense units.
end
```

of B records from the local disk. The computation time complexity of the algorithm is then $O(c^k)$, where c is a constant. The total I/O time on each processor is $O(\frac{N}{pB}k\gamma)$ as each processor has to read just $\frac{N}{p}$ part of the data in chunks of B records. The factor of k is due to the k passes required over the database before the algorithm terminates. The communication time is $O(\alpha Spk)$. The total time complexity of the algorithm is then $O(c^k + \frac{N}{pB}k\gamma + \alpha Spk)$. The running time on a single processor can simply be obtained by substituting $p = 1$ and $S = 0$, as there will be no communication.

4. Performance Evaluation

Our implementation of pMAFIA is on an IBM SP2, which is a collection of IBM RS/6000 workstations connected with a fast communication switch. In our setup of 16 processors, each processor is a 120MHz thin node with 128MB main memory, and a 2GB disk on each node of which 1GB is available as scratch space. The communication switch has a latency of 29.3 milliseconds and the bandwidth is 102 MB/sec (uni-directional) and 113 MB/sec (bi-directional). We use Message Passing Interface (MPI) for communication between processors.

4.1. Data Sets

We created a data generator to produce the data sets used in our experimental results. The data generator takes from the user the extents of the cluster in every dimension of the subspace in which it is embedded. Data can vary between any user specified maximum and minimum values for all attributes and clusters can have arbitrary shapes instead of just hyper-rectangular regions. The attribute values in the dimensions of the subspace in which the cluster is defined is generated as follows. All dimensions are scaled to lie in the range $[0, \ldots, 100]$. Data points are generated such that each unit cube, part of the user defined cluster, in this scaled space contains *at least* one point. The data so generated is scaled back appropriately into the user specified attribute ranges. This method, as against randomly populating the user defined cluster region as used in [AGGR98], ensures that we have a cluster exactly as defined by the user and helps to validate the results. For the remaining attributes we select a value at random from a uniform distribution over the entire range of the attribute. We use a better random number generator called the Inversive Congruential Generator [EHG92] as long sequences of Unix random number generators (LCGs) exhibit regular behavior by falling into specific planes. An additional 10% noise records is added to the data set. Values for all the attributes in these noise records are independently drawn at random over the entire range of the attribute. Also, user specified cluster definition is permuted to ensure no dependency on the order of input records.

4.2. Experimental Results

We present performance results of pMAFIA in terms of the speedups on different number of processors, scalability of the algorithm with the database size and scalability of the algorithm with dimensionality of the data and cluster dimensionality. Further, we compare CLIQUE with pMAFIA, followed by results of the improvement in quality of the clustering obtained by our algorithm. In the results we report below time taken for data to be read from the shared disk onto the local disks of the processors is not included as data is read over an NFS and this time varies greatly based on the network activity. We first report the results on the synthetic data sets followed by the results on real data.

Parallel Performance. Figure 4 presents the times obtained by pMAFIA. The results are on a 30 dimensional data set with 8.3 million records containing 5 clusters each in a different 6 dimensional subspace. The threshold percentages were automatically set by the algorithm for bins in all the dimensions based on the data distribution. From the plot it can be seen that we have achieved near linear speedups. This is due to the algorithm being heavily data parallel and time for computation goes down linearly with the increase in the number of processors. We observed that bulk of the time is taken in populating the candidate dense units which is completely data parallel. Communication overhead introduced due to the parallel algorithm is negligible as compared to the total time. The time taken to build the initial histogram is also a very small percentage of the total time. Further, we observed that the I/O time decreases with the increase in the number of processors as expected due to the data parallelism.

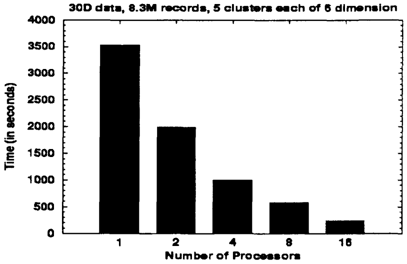


Figure 4. Parallel run times of pMAFIA

Effect of using Adaptive Grids. Figure 5 shows the speedup obtained over CLIQUE on different number of processors. The results are for a database with 300,000 records in 15 dimensions with one cluster embedded in a 5 dimensional subspace. We set the threshold percentages for the bins in various dimensions depending on the bin size based on Algorithm 1. However, while running CLIQUE we set the threshold τ to a uniform high value of 2% in all dimensions so that it could discard a larger number of candidate dense units in every pass over the data. Each dimension is divided into 10 bins for the results reported for CLIQUE. We see good parallelization for both CLIQUE and pMAFIA from the Table 1. Figure 5 shows that pMAFIA performs 40 to 80 times better than CLIQUE for this data set. The results presented for CLIQUE are using the CDU generation algorithm in [AGGR98]. This speedup is due to the minimal set of bins in each dimension based on its *interestingness*

Table 1. Execution times (in secs)

Number of Processors	1	2	4	8	16
pMAFIA	32.15	17.73	8.34	5.08	4.51
CLIQUE	2469.12	1324.51	664.65	338.19	184.36

as observed from the data histogram in every dimension. This results in a set of candidate dense units much lower than the one obtained by equal number of bins in all dimensions. Table 1 shows the comparative execution times of pMAFIA and CLIQUE for the data set used in Figure 5.

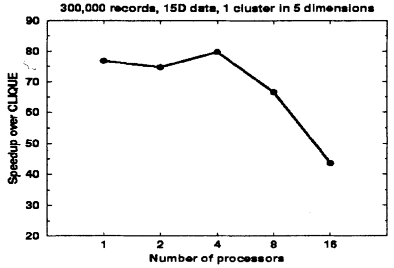


Figure 5. Speedup of pMAFIA over CLIQUE

Scalability with Database Size. Figure 6 shows the results for scalability with the database size for a 20 dimensional data with the number of records ranging from 1.45 million to 11.8 million. There were 5 clusters embedded in 5 different 5-dimensional subspaces. The results reported are on 16 processors. The thresholds for different bins were determined automatically by Algorithm 1. The time spent in cluster detection almost shows a direct linear relationship with the database size. The linear behavior is because the number of passes over the database depends only on the dimensionality of the embedded cluster. An increase in the size of the database just means that more data has to be scanned on every pass over the database while finding the dense units resulting in a linear increase in time.

Scalability with Data and Cluster Dimensions. In Figure 7 we see that pMAFIA scales very well with the increase in data dimension. The results shown are on a data set of 250, 000 records with 3 clusters each in a five dimensional subspace, with a total of 9 distinct dimensions. The results reported are on 16 processors with similar behavior observed on other number of processors. The linear behavior is due to the fact that our algorithm makes use of data distribution in every dimension and only depends on the number of distinct cluster dimensions. CLIQUE

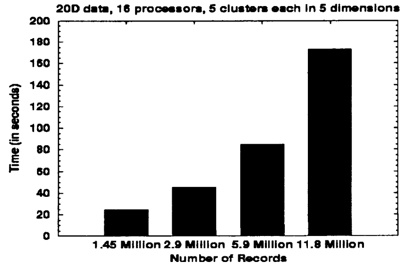


Figure 6. Scalability with database size

not only depends on distinct cluster dimensions but also on the data dimensionality. Hence it exhibits a quadratic behavior with respect to the data dimensionality as reported in [AGGR98].

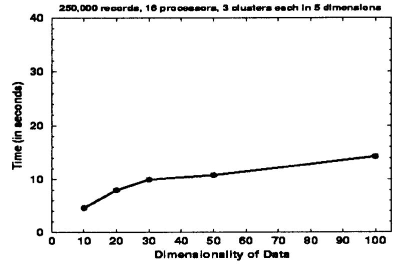


Figure 7. Scalability with Data Dimension

Figure 8 shows the scalability observed with increasing cluster dimensionality in pMAFIA. The results reported are for a 50-dimensional data set with 650,000 records containing 1 cluster on 16 processors. Results show that the time increase with cluster dimensionality reflects the time complexity of the algorithm, which is exponential in the number of distinct cluster dimensions. Similar behavior is observed on other number of processors.

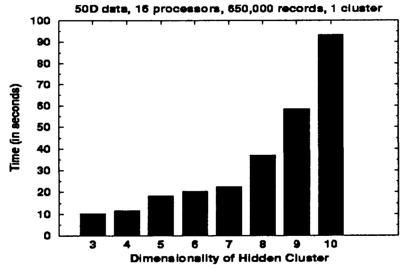


Figure 8. Scalability with Cluster Dimension

Quality of Results. We compare the quality of the results obtained by pMAFIA with those of CLIQUE, shown in Table 2. The results are for a relatively small data set with 400, 000 records in 10 dimensions with 2 clusters each in a different 4 dimensional subspace. We ran our parallelized version of CLIQUE on 16 processors. In the first case we set the number of bins to be 10 in every dimension and also set a threshold of 1% uniformly in all dimensions (as implemented in CLIQUE). In the second case we set arbitrary number of bins in each dimension (with a minimum of 5 bins to a maximum of 20 bins per dimension). The threshold in each dimension is set to 1%. In the first case CLIQUE reported the correct dimensions of the 2 clusters, however, it detected the 2 clusters only partially and large parts of the clusters were thrown away as outliers. In the second run, with a variable number of bins in each dimension, it completely failed to detect one of the clusters and, as before, the single cluster was partially detected. This is due to the inherent nature of CLIQUE which uses fixed discretization of the dimensions and hence results in a loss in the quality of the cluster obtained. For real life data sets validation of the results obtained would be a very hard task and thus bin selection would be a non trivial problem. When we ran pMAFIA on the same data set on 16 processors, both the clusters and the cluster boundaries in each dimension were accurately reported.

Table 2. Quality of Clustering

	Cluster Dimensions	Clusters Discovered
CLIQUE (fixed 10 bins)	{1,7,8,9},{2,3,4,5}	{1,7,8,9},{2,3,4,5}
CLIQUE (variable bins)	{1,7,8,9},{2,3,4,5}	{2,3,4,5}
MAFIA	{1,7,8,9},{2,3,4,5}	{1,7,8,9},{2,3,4,5}

Real Data Sets. We applied pMAFIA on three real world data sets to discover embedded clusters in different subspaces. The data sets used are of very high dimensionality. Although the data set size is small compared to the synthetic data sets used in our experiments, we report the results mainly to illustrate the applicability of pMAFIA, a completely un-supervised data mining algorithm, on real world problems. Our experiments show that for such data sets task parallelism is more effective than data parallelism. However, it is easy to see that one would gain enormously from both task and data parallelism with the increase in the data set size.

1 One Day Ahead Prediction of DAX

The data set is a one day ahead prediction of the German Stock index (DAX, Deutscher Aktien Index) based on twelve input time series which includes different stock indices, bond indices and inflation indicators like the DAX Price Index, DAX Price-Earnings Ratio and DAX Composite. Detailed explanation of the DAX data

Table 3. Clusters Discovered in DAX Data Set

Cluster Dimension	3	4	5	6
Number of Clusters Discovered	161	134	104	24

set and description of the inputs can be found in [WZ98]. The data set is in 22 dimensions with 2757 records. We choose the value of α to be 2 for the results reported. The clusters discovered are given in Table 3. The results reported are with 8 processors taking 8.16 seconds.

2 Ionosphere Data

We applied pMAFIA to the radar data that was collected by a system in Goose Bay, Labrador [BM98]. The data set is of 34 dimensions with 351 records. In the results reported on 8 processors we set α to be 2. We discovered 158 unique clusters in 3 dimensional subspaces and 32 unique clusters in 4 dimensional subspaces. However, when we increased α to 3 we discovered one single cluster in a 3 dimensional subspace. PROCLUS [APW+99] has reported two clusters one each in 31 and 33 dimensions for this data set. However, we believe that this could be in part due to an incorrect value of l, the average cluster dimensionality, chosen by the user. Further, [APW+99] also requires the user to specify k, the number of clusters in the data set which cannot be known apriori for real data sets.

3 EachMovie Recommendation Data

We applied pMAFIA to a massive data set which contained movie ratings. The data set was collected by DEC Systems Research Center over a period of 18 months. During this period 72916 users entered a total of $2,811,983$ ($\approx$ 2.8 Million) numeric ratings for 1628 different movies (films and videos). Each rating is characterized by four numbers. These numbers contain information about the user-id, movie-id, a score $(0-1)$ and a weight $(0-1)$. In this 4 dimensional data set we discovered 7 clusters all of dimension 2 in just about 28 seconds on a 400 MHz Pentium II processor. Clusters discovered by pMAFIA revealed interesting information about which set of movies were rated most by which set of users. The experiment reveals the scalability with respect to the data size of pMAFIA on real data as seen from the run times in Table 4.

Table 4. Parallel Performance.

Processors	1	2	4	8	16
Run Times (in sec)	144.86	70.47	36.86	20.35	10.18
Speed Up	1	2.06	3.93	7.11	14.23

5. Conclusions

In this chapter we presented pMAFIA, an efficient parallel algorithm for subspace clustering using a density and grid based approach with adaptive finite intervals. This performs two orders of magnitude better than CLIQUE and also improves the quality of clustering greatly as compared to both CLIQUE and PROCLUS. pMAFIA requires no user input, which makes it a totally unsupervised data mining algorithm. Experimental evaluations on a variety of synthetic and real data sets, with varying dimensionality and database sizes, show the gains in performance and quality of clusters. The use of adaptive grids in pMAFIA leads to large improvements over CLIQUE, as large as 80 times better for some data sets. Near linear speedups have also been obtained with very small communication overheads in the parallel formulation.

References

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.

[APW+99] C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, and J.S. Park. A Framework for Finding Projected Clusters in High Dimensional Spaces. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1999.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. 1998.

[DM99] I.S. Dhillon and D.S. Modha. A data-clustering algorithm on distributed memory multiprocessors. *Large-Scale Parallel KDD Systems, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[EHG92] Jurgen Eichenauer-Herrmann and Holger Grothe. A new inversive congruential pseudorandom number generator with power of two modulus. *ACM Transactions on Modeling and Computer Simulation*, 2(1):1–11, January 1992.

[EKSX96] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial

Databases with Noise. In *Proc. of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining*, 1996.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[GRS98] S. Guha, R. Rastogi, and K Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.

[JD88] A.K. Jain and R.C Dubes. *Algorithms for Clustering Data.* Prentice-Hall Inc., 1988.

[MS83] R.S Michalski and R.E. Stepp. Learning from Observation: Conceptual Clustering. *Machine Learning: An Artificial Intelligence Approach*, I:331–363, 1983.

[NH94a] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *Proc. 20th Int. Conf. on Very large Data Bases, Santiago, Chile*, pages 144–155, 1994.

[NH94b] R.T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. 20th International Conference on Very Large Databases*, 1994.

[Ols95] C.F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21, 1995.

[UF98] L.H. Ungar and D.P. Foster. Clustering methods for collaborative filtering. *AAAI Workshop on Recommendation Systems*, 1998.

[WZ98] A. S. Weigend and H. G. Zimmermann. Exploiting local relations as soft constraints to improve forecasting. *Journal of Computational Intelligence in Finance*, 6:14–23, 1998.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1996.

Chapter 20

EFFICIENT CLUSTERING OF VERY LARGE DOCUMENT COLLECTIONS

Inderjit S. Dhillon, James Fan, and Yuqiang Guan

Abstract An invaluable portion of scientific data occurs naturally in text form. Given a large unlabeled document collection, it is often helpful to organize this collection into clusters of related documents. By using a vector space model, text data can be treated as high-dimensional but sparse numerical data vectors. It is a contemporary challenge to efficiently preprocess and cluster very large document collections. In this paper we present a time and memory efficient technique for the entire clustering process, including the creation of the vector space model. This efficiency is obtained by (i) a memory-efficient multi-threaded preprocessing scheme, and (ii) a fast clustering algorithm that fully exploits the sparsity of the data set. We show that this entire process takes time that is linear in the size of the document collection. Detailed experimental results are presented — a highlight of our results is that we are able to effectively cluster a collection of 113,716 NSF award abstracts in 23 minutes (including disk I/O costs) on a single workstation with modest memory consumption.

Keywords: clustering, large document collections, hash tables, spherical k-means, vector space model

1. Introduction

Large collections of documents are becoming increasingly common. The public internet currently has more than 1.5 billion web pages, while private intranets also contain an abundance of text data. A vast amount of important scientific data appears as technical abstracts and papers. Given such large document collections it is important to organize them into structured ontologies. This organization facilitates navigation and search, and at the same time provides a framework for continual maintenance as document repositories grow in size.

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 357–381.
© 2001 *Kluwer Academic Publishers.*

Manual construction of structured ontologies is one possible solution and has been adopted to organize the internet (www.yahoo.com) and to structure library content. However, this process has the obvious disadvantage of being too labor intensive, and is viable only in large corporations. Thus it is desirable to seek automatic methods for organizing unlabeled document collections. Given a collection of unlabeled data points, *clustering* refers to the problem of automatically assigning class labels to the data and has been widely studied in statistical pattern recognition and machine learning [DH73, Mit97].

A starting point for applying clustering algorithms to unstructured text data is to create a *vector space model*, alternatively known as a *bag-of-words model* [SM83]. The basic idea is (a) to extract unique content-bearing words from the set of documents treating these words as *features* and (b) to then represent each document as a vector of certain weighted word frequencies in this feature space. Observe that we may regard the vector space model of a text data set as a *word-by-document matrix* whose rows are words and columns are document vectors. Typically, a large number of words exist in even a moderately sized set of documents where a few thousand words or more are common. Thus for large document collections, both the row and column dimensions of the matrix are quite large. However, as we will discuss later in greater detail, this matrix is typically very sparse with almost 99% of the matrix entries being zero.

Using the vector space model various classical clustering algorithms such as the k-means algorithm and its variants, hierarchical agglomerative clustering, and graph-theoretic methods have been explored in the text mining literature; for detailed reviews, see [Ras92, Wil88]. Recently, there has been a flurry of activity in this area, see [BGG$^+$98, CKPT92, SS97, ZE98]. A substantial amount of this work has concentrated on clustering web search results where the document collections to be clustered are not very large.

In this paper, our main concern is in obtaining a highly efficient process for clustering very large document collections. Our main motivation is that we want to cluster collections in excess of more than 100,000 documents in a reasonable amount of time on a single processor. Thus our main emphasis is on high speed and scalability with modest main memory consumption. The clustering process involves reading the text documents from disk, and preprocessing them to form the vector space model before using a particular clustering algorithm. It turns out that main memory consumption and disk I/O costs in this process can be prohibitive. In order to alleviate these problems, we employ a memory-efficient multi-threaded approach to reading and preprocessing the documents. For creating the vector space model, we use efficient and scalable data structures such as local and global hash tables. Finally, we use the highly efficient and effective *spherical k-means* algorithm that fully exploits the sparsity of the data [DM01]. The above steps lead to a highly efficient clustering process — as a result we are able to preprocess and cluster a collection of 113,716 NSF award abstracts documents in 23 minutes on a Sun workstation with modest memory consumption.

This paper is organized as follows. In Section 2 we highlight the challenges in obtaining an efficient and effective process for clustering large document collections. Section 3 describes our multi-threaded algorithm for creating the vector space model. In Section 4 we present the spherical k-means algorithm which is ideally suited for clustering high-dimensional sparse text data. Section 5 presents speed and memory consumption results on some document collections, in particular we examine the results on a collection of 113,716 NSF award abstracts. Finally, Section 6 concludes with a short summary and discussion of future work.

A quick word about notation. Lower case bold letters such as $\mathbf{x}$, $\mathbf{c}$ will denote vectors while $\|\mathbf{x}\|$ will denote the 2-norm of the corresponding vector. Thus when we say that $\|\mathbf{x}\| = 1$ we mean that $\mathbf{x}^T\mathbf{x} = 1$.

2. Challenges

As mentioned above, the preprocessing phase leads to the creation of the vector space model. Given a large document collection, the following steps are involved in this phase: (a) read all input documents from disk, (b) parse into word tokens using efficient regular expression pattern matching, (c) lookup words rapidly in hash tables to track the number of occurrences of each word, and finally, (d) output the vector space model.

Efficient parsing using regular expression pattern matching is a well-studied problem. Existing software tools such as FLEX provide a quick and easy way to construct the required parser [Pax96]. Also, efficient hash table indexing and access is provided in public-domain software [MS96]. Thus effective solutions to steps (b) and (c) above are easy to obtain.

However I/O costs in steps (a) and (d) can be substantial, especially if the documents need to be accessed from a Network File System(NFS) [Cal99]. In such a case, the time taken by these steps can be quite large and often unpredictable depending on other traffic on the NFS. Our approach to solve this problem is to use multiple threads [NBF96] so that input documents can be read in a parallel fashion.

Another challenge is main memory consumption. After all the documents have been read and processed there is a final phase which involves a sweep and modification of the entire vector space model. One simple approach to facilitate this phase is to retain the partially formed vector space model in main memory while the rest of the documents are being processed. We implemented such an approach and on our test collection of 113,716 documents, this required nearly 500 MBytes of main memory. Extrapolating this memory requirement to 1 million documents, we can see that this main-memory based implementation will require several gigabytes of memory. This cost is unacceptable since our goal is to preprocess and cluster a million documents on current workstations consuming less than 256 MBytes of main memory. In Section 3 we introduce such a memory efficient algorithm that allows us to preprocess a large number of documents without sacrificing much speed.

In our test collection of 113,716 documents there are a total of more than 150,000 unique words. After a pruning step, we retain 26,000 unique words in the vector space model. Thus, the document vectors are very *high-dimensional*. However, typically, most documents contain only a small subset of the total number of words. Hence, the document vectors are very *sparse* — a sparsity of 99% is common. The major challenge is to find a clustering algorithm that can yield good effective solutions for very high-dimensional data and at the same time, exploit the sparsity of the vector space model. Since we are interested in clustering very large document collections we seek algorithms that consume time and memory that is linear in the size of the document collection.

3. Efficient Preprocessing

In this section, we describe our preprocessing algorithm for creating the vector space model. Before we describe the details we give a high level description of the task at hand.

3.1. Vector Space Model

The basic idea is to represent each document as a vector of certain weighted word frequencies. In order to do so, the following parsing and extraction steps are needed.

1 Ignoring case, extract all unique words from the entire set of documents.

2 Eliminate non-content-bearing "stopwords" such as "a", "and", "the", etc. For sample lists of stopwords, see [FBY92, Chapter 7].

3 For each document, count the number of occurrences of each word.

4 Using heuristic or information-theoretic criteria, eliminate non-content-bearing "high-frequency" and "low-frequency" words [SM83].

5 After the above elimination, suppose w unique words remain. Assign a unique identifier between 1 and w to each remaining word, and a unique identifier between 1 and d to each document.

The above steps outline a simple preprocessing scheme. In addition, one may extract word phrases such as "New York," and one may reduce each word to its "root" or "stem", thus eliminating plurals, tenses, prefixes, and suffixes [FBY92, Chapter 8].

The above preprocessing yields the number of occurrences of word j in document i, say, f_{ji}, and the number of documents which contain the word j, say, d_j. Using these counts, we can represent the i-th document as a w-dimensional vector $\mathbf{x}_i$ as follows. For $1 \leq j \leq w$, set the j-th component of $\mathbf{x}_i$, to be the product of three terms

$$x_{ji} = t_{ji} \cdot g_j \cdot s_i,$$

where t_{ji} is the *term weighting component* and depends only on f_{ji}, while g_j is the *global weighting component* and depends on d_j, and s_i is the *normalization component* for $\mathbf{x}_i$. Intuitively, t_{ji} captures the relative importance of a word in a document, while g_j captures the overall importance of a word in the entire set of documents. The objective of such weighting schemes is to enhance discrimination between various document vectors for better retrieval effectiveness [SB88].

There are many schemes for selecting the term, global, and normalization components, see [Kol97] for various possibilities. In this paper we use the popular tfn scheme known as *normalized term frequency-inverse document frequency*. This scheme uses $t_{ji} = f_{ji}$, $g_j = \log(d/d_j)$ and $s_i = \left(\sum_{j=1}^{w}(t_{ji}g_j)^2\right)^{-1/2}$. Note that this normalization implies that $\|\mathbf{x}_i\| = 1$, i.e., each document vector lies on the surface of the unit sphere in R^w. Intuitively, the effect of normalization is to retain only the *proportion* of words occurring in a document. This ensures that documents dealing with the same subject matter (that is, using similar words), but differing in length lead to similar document vectors.

3.2. Preprocessing Algorithm

The input to the preprocessing step is a data directory that contains all the documents to be processed. The documents may also be contained within subdirectories of the input directory. The output is the vector space model described in the above section, which can be represented as

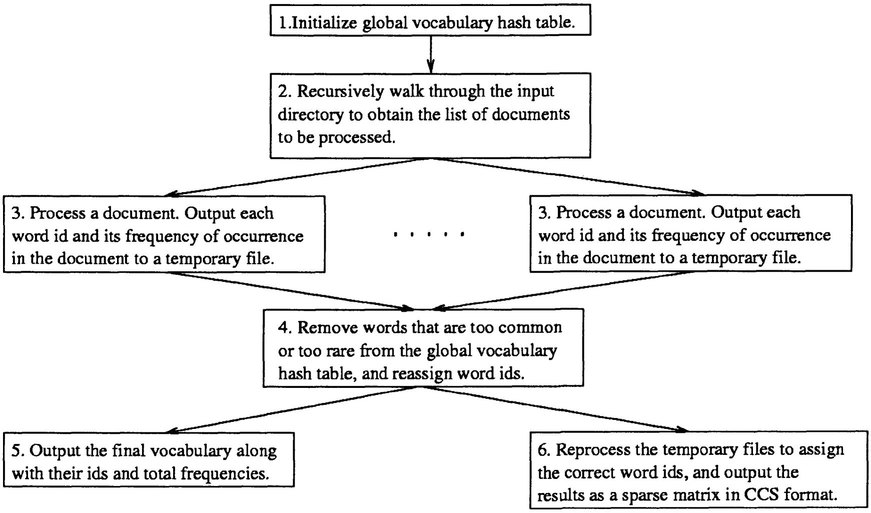

1.Initialize global vocabulary hash table.
2. Recursively walk through the input directory to obtain the list of documents to be processed.
3. Process a document. Output each word id and its frequency of occurrence in the document to a temporary file.
. . . . .
3. Process a document. Output each word id and its frequency of occurrence in the document to a temporary file.
4. Remove words that are too common or too rare from the global vocabulary hash table, and reassign word ids.
5. Output the final vocabulary along with their ids and total frequencies.
6. Reprocess the temporary files to assign the correct word ids, and output the results as a sparse matrix in CCS format.

Figure 1. Outline of the preprocessing algorithm

a highly sparse word-by-document matrix. We store this sparse matrix by using the Compressed Column Storage(CCS) format [DGL89]. In this format, we record the value of each non-zero element, along with its row and column index. The column indices represent the input documents, the row indices represent ids of distinct words present in the document collection, and the non-zero entries in the matrix represent the frequencies of words in documents.

Figure 1 gives an outline of the preprocessing algorithm. The algorithm first initializes a global hash table. To resolve whether a word has been encountered previously, a local and a global hash table are used. Both these hash tables use words as keys and store the corresponding row indices and frequencies as values. As the names suggest, the global hash table keeps track of words and their occurrences in the entire document collection while the local hash table does so for just one document. After initializing the global hash table, the algorithm recursively walks through the input directory to obtain the list of documents to be processed. The preprocessing algorithm then creates several threads of computation. The purpose of each thread is to process a set of documents independently and output its results into temporary files. Details of this processing are given in Figure 2 which we discuss in the next paragraph. After all the threads have finished, the global hash table is examined, and words that are too common or too rare are removed from the global hash table. Unique word ids are assigned to the words

1. Initialize the local hash table (this hash table uses words as keys, and stores row indices and frequencies as values).
2. Get a token from the document, and convert the token to lowercase.
3. Discard the token if it is too long or too short or is a non-content "stopword" such as "a", "and", "the", etc.
4. If the word already exists in the local hash table, increment the frequency of that entry, otherwise insert the word into the local hash table with row index −1 and frequency 1.
5. After the whole document has been processed, set the row indices in the local hash table to the corresponding ones in the global hash table. If a word in the local hash table does not exist in the global hash table, assign a new row index to the word and add to the global hash table (note that this requires locking the global hash table to prevent simultaneous modification by another thread).
6. Output the contents of the local hash table (row indices and frequencies) to a temporary file, and discard the local hash table.

Figure 2. Details on the processing of a document by each thread (step 3 of Figure 1)

that still remain in the global hash table. The temporary files are then reloaded, the word ids are resolved and then the final vocabulary and word-by-document matrix are output.

Figure 2 describes the various steps performed by each preprocessing thread. Two decisions warrant further explanation — the use of temporary files for storing the partial vector space model and the way in which the local and global hash tables are accessed. As mentioned in the last section, storing the partial vector space model in main memory would require a few gigabytes of main memory and is thus prohibitive for modern workstations. Hence to reduce main memory consumption we store the contents of the local hash table onto temporary files. Since this only leads to local disk access, the resulting overhead is not substantial.

The global hash table is accessed and modified by all processing threads and hence is a shared resource. In order to achieve maximum parallelism, we need to minimize the number of times the global hash table is locked and modified by each processing thread. We achieve this by using a local hash table to process the entire document first, and then making a block access to the global hash table. This access involves resolving the word ids and possible modification of the global hash table, at which time this data structure needs to be locked. See Figure 2 for details, especially step 5.

The major main memory consumption in our preprocessing algorithm is due to the global vocabulary table. The partial vector space model is stored in temporary files instead of main memory. Thus the overall main memory requirement is $O(W)$, where W is the number of distinct words that appear in the document collection. It is well known that W grows slower than linearly with the size of the document collection — ***Heaps' Law*** states that the number of unique words in a text of size d grows as $O(d^\beta)$, where β is a positive number less than one[Hea78]. Thus the main memory consumption grows slower than linearly with the size of the document collection. The computation time for the preprocessing step is approximately linear with respect to the input data size since each word in a document is processed in $O(1)$ amortized time. Performance results on large document collections that validate these claims are given in Section 5.

4. Efficient Clustering of High-Dimensional Text Data

Given the vector space model, the document vectors may be represented by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d$, where each $\mathbf{x}_i \in R^w$. Recall that w stands for the number of unique words in the vector space model and d is the total number of documents. A clustering of the document collection is its partitioning into the disjoint subsets $\pi_1, \pi_2, ..., \pi_k$, i.e.,

$$\bigcup_{j=1}^{k} \pi_j = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_d\} \quad \text{and} \quad \pi_j \cap \pi_l = \phi, \;\; j \neq l.$$

The most important and challenging characterisitics of the vector space models that arise from text data are high dimensionality and sparsity. Typically, w is in the thousands and a sparsity of 99% is common. For purposes of efficiency, it is important that the clustering algorithm exploit the sparsity of the data while giving meaningful results at the same time. The spherical k-means algorithm satisfies both these properties and hence is our algorithm of choice. We now briefly formalize this algorithm highlighting its salient features. More details may be found in [DM01].

Any text clustering algorithm needs an objective notion of similarity between documents. A widely used measure of similarity is the cosine of the angle between two document vectors [FBY92, SM83]. Cosine similarity is easy to interpret and simple to compute for sparse vectors and has been used in other information retrieval applications, such as query retrieval. Based on cosine similarity, we can define the "goodness" or "coherence" of cluster π_j as

$$\sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i^T \mathbf{c}_j, \tag{4.1}$$

where each $\mathbf{x}_i$ is assumed to be normalized such that $\|\mathbf{x}_i\| = 1$ and $\mathbf{c}_j$ is the normalized centroid of cluster π_j,

$$\mathbf{c}_j = \frac{\sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i}{\|\sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i\|}.$$

By the Cauchy-Schwarz inequality,

$$\sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i^T \mathbf{z} \le \sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i^T \mathbf{c}_j, \qquad \forall \mathbf{z} \in R^w,$$

and thus the normalized centroid is the vector that is closest in cosine similarity (in an average sense) to all the document vectors in the cluster π_j. As a result, we also call the vector $\mathbf{c}_j$'s as *concept vectors.*

Aggregating (4.1) over all clusters, we can measure the goodness of any given partitioning $\{\pi_j\}_{j=1}^k$ using the following *objective function*:

$$Q(\{\pi_j\}_{j=1}^k) = \sum_{j=1}^k \sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i^T \mathbf{c}_j. \tag{4.2}$$

Intuitively, the objective function measures the combined coherence of all the k clusters. Having posed the objective function, we now present an algorithm that attempts to maximize its value.

4.1. Spherical k-means Algorithm

It is well known that finding the partitioning that maximizes (4.2) is NP-Complete [KPR98, Theorem 3.1]; also, see [GJW82]. Thus we seek a heuristic algorithm that can quickly find a good local maximum. For this purpose, we use a variant of the classical k-means algorithm [DH73] that uses the cosine similarity measure. Since both the document and concept vectors lie on the surface of a high-dimensional sphere, we call this variant the *spherical k-means* algorithm. This algorithm proceeds as follows.

1. Initialize clustering. Start with some initial partitioning of the document vectors, namely $\{\pi_j^{(0)}\}_{j=1}^{k}$. Let $\{\mathbf{c}_j^{(0)}\}_{j=1}^{k}$ be the concept vectors of the associated partitioning. Set the iteration count t to 0.

2. Re-assign document vectors. For each document vector $\mathbf{x}_i$, $1 \leq i \leq d$, do the following:

   a. Compute $\mathbf{x}_i^T \mathbf{c}_l^{(t)}$ for all $l = 1, 2, \ldots, k$.

   b. From among all $\mathbf{x}_i^T \mathbf{c}_l^{(t)}$ computed above, find $j = \arg\max_l \mathbf{x}_i^T \mathbf{c}_l^{(t)}$ (break ties arbitrarily if $\mathbf{x}_i$ has largest cosine similarity with more than one concept vector).

   This induces the new partitioning

   $$\pi_j^{(t+1)} = \{\mathbf{x}_i : j = \arg\max_l \mathbf{x}_i^T \mathbf{c}_l^{(t)}\}, \quad 1 \leq j \leq k.$$

3. Update concept vectors. Compute the concept vectors corresponding to the new partitioning:

   $$\mathbf{s}_j = \sum_{\mathbf{x}_i \in \pi_j} \mathbf{x}_i, \qquad \mathbf{c}_j^{(t+1)} = \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}, \quad 1 \leq j \leq k.$$

4. Check the stopping criterion. If the stopping criterion is satisfied, then exit. Otherwise increase t by 1 and go to step 2 above.

In our implementation, the stopping criterion is:

$$|Q(\{\pi_j^{(t)}\}_{j=1}^{k}) - Q(\{\pi_j^{(t+1)}\}_{j=1}^{k})| \leq 10^{-3} \cdot |Q(\{\pi_j^{(t)}\}_{j=1}^{k})|.$$

It can be shown that the above algorithm yields a gradient-ascent scheme. In particular, we can show that the objective function value in (4.2) does not decrease from one iteration to the next, i.e.,

$$Q(\{\pi_j^{(t)}\}_{j=1}^{k}) \leq Q(\{\pi_j^{(t+1)}\}_{j=1}^{k}).$$

Like any other gradient-ascent scheme, the spherical k-means algorithm is prone to local maxima. A careful selection of initial partitions

$\{\pi_j^{(0)}\}_{j=1}^k$ is important. One can either (a) randomly assign each document to one of the k clusters, (b) first compute the concept vector for the entire document collection and randomly perturb this vector to get k starting concept vectors or (c) try several initial clusterings and select the best in terms of the largest objective function. We use strategy (b) in our implementation.

We now examine the time and memory complexity of the above algorithm. We assume that the number of non-zero entries in the sparse matrix is nz, and that the above algorithm iterates τ times before stopping. Using our initialization strategy, step 1 of the algorithm costs $O(nz + k \cdot w)$ operations. For each iteration, step $2a$ costs $O(nz \cdot k)$ operations while $2b$ costs $O(k \cdot d)$ comparisons. Step 3 updates the concept vectors and costs $O(nz + k \cdot w)$ operations. Thus the total time complexity for τ iterations is

$$O((nz \cdot k + k \cdot d + k \cdot w) \cdot \tau).$$

Typically $nz \gg \max(d, w)$, hence it is clear that step $2a$ is the most computationally expensive step in the algorithm and the overall complexity of the algorithm is $O(nz \cdot k \cdot \tau)$.

The main memory consumed by the algorithm is for storing the document vectors and for old and new copies of the concept vectors. Storing the document vectors in the CCS sparse matrix form requires $4(2nz+d+4)$ bytes while the concept vectors require $8kw$ bytes; hence the memory consumption is modest.

4.2. Similarity Estimation

The computational bottleneck in the spherical k-means algorithm is the dot product computation between all the document vectors and concept vectors (see step 2a). During the course of the algorithm it turns out that the first few iterations lead to a lot of movement of documents between clusters. However, just after a few iterations the clusters become more stable, see the solid line in the left plot of Figure 3. Consequently, the overall objective function value also settles down after 4-5 iterations as seen in the right plot of Figure 3.

When the clusters become stable, there are potential savings if we can somehow avoid computing unnecessary dot products between document vectors and 'far away' concept vectors. We now introduce a technique for estimating cosine similarities which allows us to avoid such dot product computations.

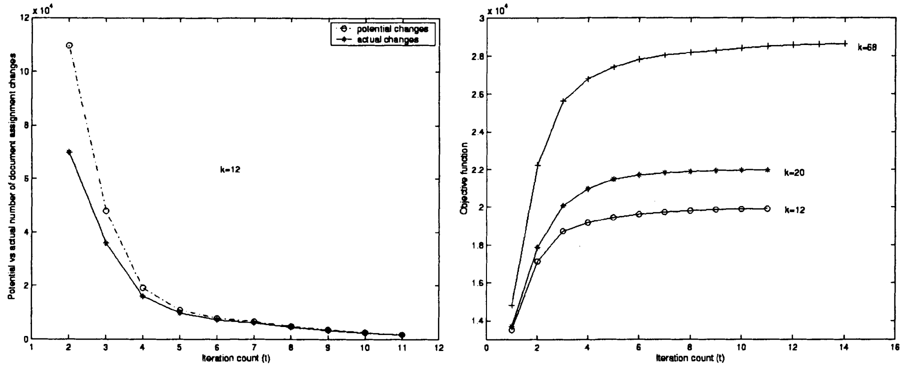


Figure 3. Clusters stabilize after just a few iterations.

Suppose $\mathbf{c}_l^{(t)}$ is the concept vector of cluster l at iteration t and $\mathbf{x}$ is a document vector. Then,

$$|\mathbf{x}^T\mathbf{c}_l^{(t)} - \mathbf{x}^T\mathbf{c}_l^{(t+1)}| \leq \|\mathbf{x}\|\|\mathbf{c}_l^{(t)} - \mathbf{c}_l^{(t+1)}\| = \|\mathbf{c}_l^{(t)} - \mathbf{c}_l^{(t+1)}\|,$$

$$\Rightarrow \mathbf{x}^T\mathbf{c}_l^{(t)} - \|\mathbf{c}_l^{(t)} - \mathbf{c}_l^{(t+1)}\| \leq \mathbf{x}^T\mathbf{c}_l^{(t+1)} \leq \mathbf{x}^T\mathbf{c}_l^{(t)} + \|\mathbf{c}_l^{(t)} - \mathbf{c}_l^{(t+1)}\| \quad (4.3)$$

The right side of inequality (4.3) gives a similarity upper bound which can be used profitably in the $(t+1)$-st iteration to avoid computing $\mathbf{x}^T\mathbf{c}_l^{(t+1)}$. The idea is to store in memory the quantities $\|\mathbf{c}_l^{(t)} - \mathbf{c}_l^{(t+1)}\|$ and upper bounds for $\mathbf{x}^T\mathbf{c}_l^{(t)}$. Suppose $\mathbf{x}$ belongs to the cluster j at the t-th iteration. Then, at iteration $t+1$, we can update in $O(1)$ time the similarity upper bound for $\mathbf{x}^T\mathbf{c}_l^{(t+1)}$, $l \neq j$. If this similarity upper bound is smaller than $\mathbf{x}^T\mathbf{c}_j^{(t+1)}$ (which is computed exactly) there is no need to explicitly compute $\mathbf{x}^T\mathbf{c}_l^{(t+1)}$, otherwise the exact value needs to be computed. As the clusters become more and more stable, this similarity estimation can dramatically cut down on computation (see Figure 4).

In our implementation, we store the similarity upper bounds in the $d \times k$ matrix U. We obtain the new algorithm by replacing step 2 of the spherical k-means algorithm by the following step (here we assume that similarity estimation is started after t_{min} iterations).

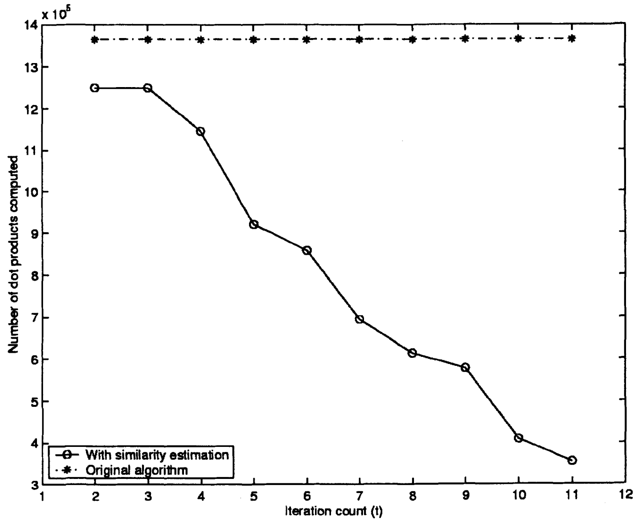


Figure 4. Computational Savings due to Similarity Estimation

```
2. Re-assign document vectors. For each document vector $\mathbf{x}_i$, $1 \le i \le d$,
   do the following:

   a. If $t \le t_{min}$, do step 2a as in Section 4.1.
      else if $t = t_{min}$, set $U(i,l) = \mathbf{x}_i^T \mathbf{c}_l^{(t)}$ for all $l = 1, 2, \dots, k$.
      else if $t > t_{min}$, do the following steps for all $l = 1, 2, \dots, k$,
      a1. Set $U(i,l) = U(i,l) + \|\mathbf{c}_l^{(t)} - \mathbf{c}_l^{(t-1)}\|$.
      a2. Compute $\mathbf{x}_i^T \mathbf{c}_j^{(t)}$ where $\mathbf{x}_i$ belongs to cluster $j$ at iteration $t-1$.
          If $U(i,l) > \mathbf{x}_i^T \mathbf{c}_j^{(t)}$, compute $\mathbf{x}_i^T \mathbf{c}_l^{(t)}$ and set $U(i,l) = \mathbf{x}_i^T \mathbf{c}_l^{(t)}$.

   b. From among all $\mathbf{x}_i^T \mathbf{c}_l^{(t)}$ computed above, find $j = \arg\max_l \mathbf{x}_i^T \mathbf{c}_l^{(t)}$
```

Figure 4 shows the considerable savings in the number of dot product computations as the iteration count increases in a typical run of the algorithm. To obtain these savings, an extra storage requirement of $4dk$ bytes is required to store the matrix U. Additionally, an extra $O(dk)$ operations to update the matrix U are required (see step a1 above).

However, these extra costs are small compared to the resulting savings in computation time, see Section 5.1.2 for an example.

4.3. Clustering for Dimensionality Reduction

We now highlight another use of our clustering algorithm. Using the vector space model, each document may be represented as a high-dimensional vector of words. Clearly the occurrence of one word in a document is not independent of other words. Thus there is a great deal of redundancy in this collection of vectors. Dimensionality reduction is a technique that tries to represent each document as a vector with fewer dimensions that are more independent. It turns out that the above clustering algorithm also yields a fast and effective technique for dimensionality reduction.

Let $\{\mathbf{c}_j\}_{j=1}^{k}$ denote the concept vectors corresponding to a clustering of the document vectors. The *concept matrix* C_k is defined to be a $d \times k$ matrix such that, for $1 \leq j \leq k$, the j-th column of the matrix is the concept vector $\mathbf{c}_j$, i.e,

$$C_k = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k].$$

An approximation X_k to the word-document matrix X may be obtained by taking the least squares projection of X onto the column space of C_k, i.e., the linear subspace spanned by the concept vectors. This may be expressed as

$$X_k = C_k Z_k^*,$$

where Z_k^* is a $k \times d$ matrix that is to be determined by solving the following least-squares problem:

$$Z_k^* = \arg\min_{Z} \|X - C_k Z\|_F^2.$$

It is well known that a closed form solution exists for this least-squares problem, namely,

$$Z_k^* = (C_k^T C_k)^{-1} C_k^T X. \tag{4.4}$$

The i-th column of Z_k^* gives the reduced k-dimensional representation of the i-th document vector. Typically the original dimensionality w is in the thousands while k is much smaller. In previous work, we have shown that empirically, the quality of the reduced dimensional representation given by (4.4) is comparable to the "best" possible, namely the k-truncated SVD, see [DM01] for details. Thus our clustering process followed by dimensionality reduction can be used in various applications, such as query retrieval, text classification, etc.

5. Experimental Results

In this section, we give experimental results for the entire clustering process — this includes time and memory consumed by the preprocessing phase as well as by the clustering algorithm. In addition, we need to evaluate the goodness of the clustering produced.

Evaluating clustering results is a tricky business. However, in situations where documents are already categorized(labelled), we can compare the clusters with the "true" class labels. For this comparison we use the measures of purity and entropy as defined below, see also [SGM00].

Suppose we are given c categories (true class labels) while the clustering algorithm produces k clusters. Cluster π_l's *purity* can be defined as

$$P(\pi_l) = \frac{1}{n_l} \max_h (n_l^{(h)}),$$

where $n_l = |\pi_l|$ and $n_l^{(h)}$ is the number of documents in π_l that belong to class h, $h = 1, \ldots, c$. Note that each cluster may contain samples from different classes. Purity gives the ratio of the dominant class size in the cluster to the cluster size itself. A high purity value implies that the cluster is a "pure" subset of the dominant class.

Additionally, we also use *entropy* as a quality measure, which is defined as follows:

$$H(\pi_l) = -\frac{1}{\log c} \sum_{h=1}^{c} \frac{n_l^{(h)}}{n_l} \log \left(\frac{n_l^{(h)}}{n_l} \right).$$

Entropy is a more comprehensive measure than purity. It considers the distribution of classes in a cluster. Note that we have normalized entropy to take values between 0 and 1. An entropy value of 0 means the cluster is comprised entirely of one class, while an entropy value near 1 is bad since it implies that the cluster contains a uniform mixture of classes.

We now examine these quality measures on a sample collection. We formed a collection of 3893 documents by merging the popular MEDLINE, CISI, and CRANFIELD sets. MEDLINE consists of 1033 abstracts from medical journals, CISI consists of 1460 abstracts from information retrieval papers, while CRANFIELD consists of 1400 abstracts from aeronautical systems papers (`ftp://ftp.cs.cornell.edu/pub/smart`). We preprocessed this collection by proceeding as in Section 3. After removing common stopwords, the collection contained 22149 unique words from which we eliminated 17839 low-frequency words appearing in less than 8 documents (roughly 0.2% of the documents), and 7 high-frequency words appearing in more than 585 documents (roughly 15% of the documents). We were finally left with 4303 words using which we created 3893 document vectors using the tfn scheme. Each document vector has dimension 4303, however, on an average, each document vector is about 99% sparse.

For this contrived collection, we used our clustering algorithm to form 3 clusters. The following table shows the "confusion matrix" for this clustering, from which we can see that the clusters can be easily identified with the three classes — 'MEDLINE', 'CISI' and 'CRANFIELD'.

	MEDLINE	CRANFIELD	CISI
patients(1023)	**1020**	2	1
boundary(1388)	1	**1383**	4
library (1481)	12	14	**1455**

In the above table the rows denote clusters. Note that we have denoted the clusters by the most frequently occurring word in the cluster from among the words in the vector space model — 'patients', 'boundary' and 'library'. The above table says that the cluster 'patients' has 1023 documents of which 1020 belong to the 'MEDLINE' class, the 'boundary' cluster has 1388 documents while the 'library' cluster has 1481 documents. Notice that the confusion matrix is almost diagonal which shows that the clustering algorithm nearly recovered the three classes. This fact is also reflected by the high purity and low entropy values of the three clusters shown below.

Cluster#	Purity	Entropy
0	.996094	.0252647
1	.996398	.0233619
2	.982444	.0914646

5.1. Case Study on a Large Scientific Collection

To show the efficiency and effectiveness of our algorithms, we now present results on a large real-life collection of NSF award abstracts. We obtained the NSF data set by downloading abstracts of grants awarded by the National Science Foundation between Jan 1958 and August 1999 from www.nsf.gov. For our experiments we extracted titles and abstracts (ignoring fields such as 'Type', 'NSF Org', 'Date', etc.) of awards made in the 8 NSF organizations: Directorate for Biological Sciences (BIO), Directorate for Computer and Information Science and Engineering (CISE), Directorate for Education and Human Resources (EHR), Directorate for Engineering (ENG), Directorate for Geosciences (GEO), Directorate for Mathematical and Physical Sciences (MPS), Office of

Polar Programs (OPP) and Directorate for Social, Behavioral, and Economic Sciences (SBE). This results in a total of 113716 abstracts. The number of documents per NSF organization is as follows:

Class label	ENG	CSE	MPS	BIO	GEO	EHR	SBE	OPP
# documents	19365	8275	23037	16768	15667	14112	13725	2767

The most populous class is MPS with over 23000 awards, with ENG being second. On the other hand, OPP contains the fewest (2767) number of documents. Both the mean and median class size is about 142000.

We preprocessed this NSF collection by proceeding as in Section 3. After removing common stopwords, the collection contained 153527 unique words from which we eliminated 127103 low-frequency and high-frequency words. We were finally left with 26424 words using which we created 113716 document vectors using the tfn scheme. Each document vector has dimension 26424, however, on an average, each document vector contains only about 72 nonzero components and thus is more than 99% sparse.

In terms of the words contained in these documents, Table 1 shows the top ten most common words in the NSF data set. Note that we refer to the most frequently occurring word as having rank 1, the second most frequent word as having rank 2, and so on. As is common in most document collections the majority of the words occur in very few documents. Figure 5 shows the distribution of all the word frequencies versus their rank on a log-log scale. This distribution approximately fits the so-called *Zipf's law* [Zip49].

Rank	Word	Frequency
1	abstract	154467
2	research	152134
3	title	143257
4	project	73720
5	study	51178
6	data	43703
7	system	40314
8	university	39491
9	program	39203
10	science	38530

Table 1. Top 10 most common words in the NSF data set

As mentioned above, there are a total of 153527 unique word in the NSF collection. In general, the number of unique words *W* grows slower

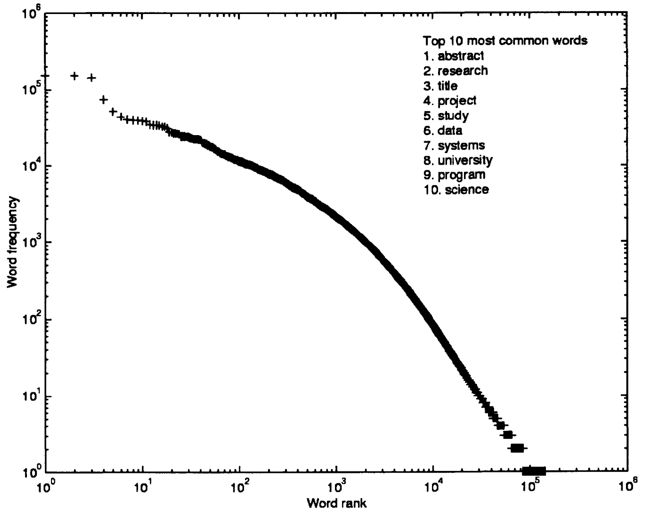


Figure 5. Distribution of word frequencies on a log-log scale (Zipf's Law)

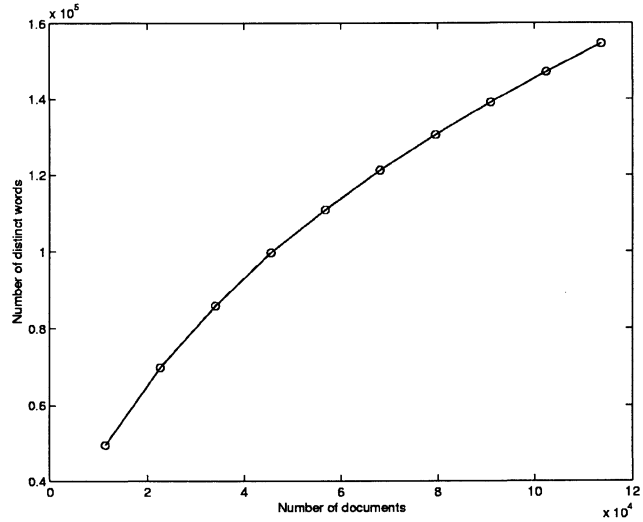


Figure 6. Vocabulary size vs. number of documents (Heap's Law)

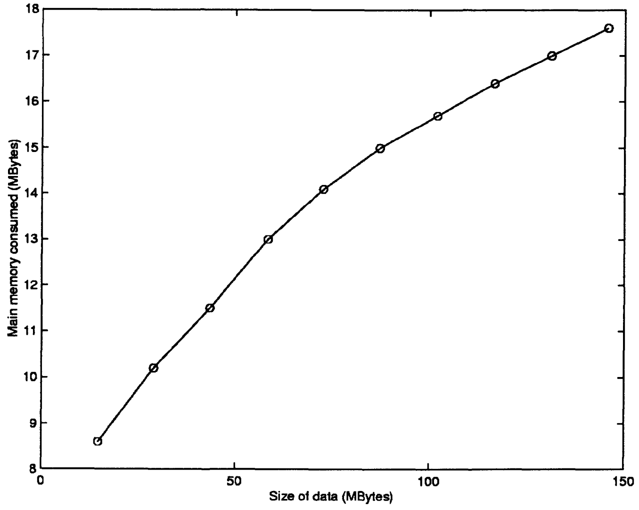


Figure 7. Main memory consumed vs. size of data

than linearly with the number of documents, i.e., $W = O(d^{\beta})$ where $0 < \beta < 1$. This behaviour known as *Heap's law* [Hea78] is seen in Figure 6.

5.1.1 Preprocessing Results.

We now show the results of our clustering process on this large NSF collection. The preprocessing phase takes time that grows linearly with the size of the data set, while the main memory consumed also grows linearly with the number of unique words. Experiments on subsets of the NSF collection validate these claims, see Figures 7 and 8.

Figure 7 shows that we consume less than 18 MBytes of main memory to preprocess the entire NSF collection of 113716 documents. Extrapolating this number to a collection of one million documents, we see that 160 MBytes of main memory would be sufficient, which implies that our algorithms can carry out this large task on a single workstation with just 256 MBytes of main memory. From Figure 8 we see that the entire NSF data set is preprocessed in less than 20 minutes. Note that this includes disk I/O time. Our implementation is multi-threaded and hence the time taken is not very sensitive to the traffic over the Network File Server.

5.1.2 Clustering Results.

In this section, we examine the speed and quality of the clustering algorithm. As discussed in Sec-

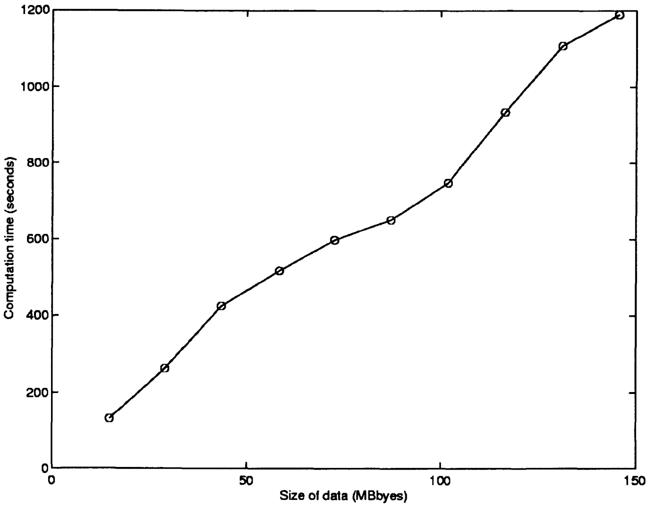


Figure 8. Computation time vs. size of data

tion 4.1, the spherical k-means algorithm forms k clusters in $O(nz \cdot k)$ time. Figure 9 gives the time taken to cluster the NSF data set (113716 documents, 26424 words, 8141987 non-zero entries) into a varied number of clusters. Times for both the original clustering algorithm and its modification that uses similarity estimation are shown (see Section 4.2). The similarity estimation technique yields considerable savings in computation time when the number of clusters is large. The running time of the algorithm is also seen to increase linearly with the number of clusters k. To form 100 clusters, 1400 seconds of computational time is needed. On the other hand, to form 12 clusters only 200 seconds are needed. Thus, combined with the preprocessing time of about 1190 seconds, the entire clustering process for 12 clusters is seen to take only about 23 minutes.

We now present a sample clustering obtained by the spherical k-means algorithm. We clustered the entire set of 113716 documents into 12 clusters. In the following table, we have listed the 10 dominant words and their weights in each of the concept vectors that correspond to the 12 clusters. Note that we have denoted each of the 12 clusters by the dominant word, e.g., 'theory', 'chemistry', 'physics', etc.

By examining the top 10 words it is easy to see the 'concept' contained in each cluster. For example, the 'conference' cluster containing 6652 documents appears to be about NSF awards that support conferences, meetings, workshops and symposiums for international scientists and researchers.

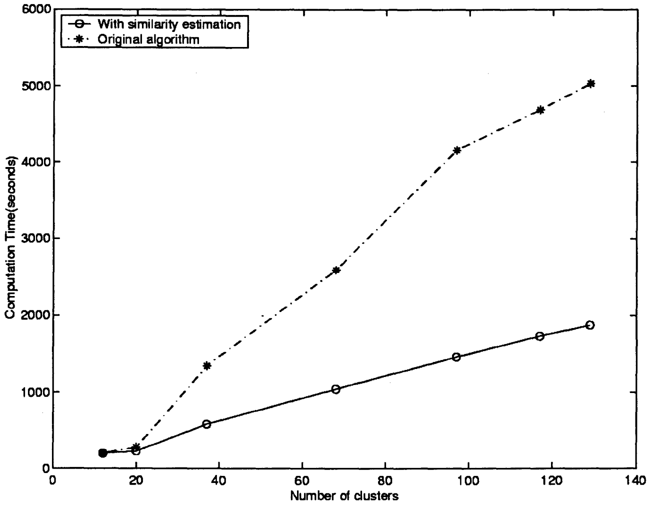


Figure 9. Time required to cluster the NSF collection

We now see the extent to which our clustering algorithm is able to recover the 8 NSF classes: BIO, CSE, EHR, ENG, GEO, MPS, OPP and SBE. The following gives the confusion matrix between the 12 clusters and 8 classes.

Note that the largest number in each row of the above table is boldfaced. We have ordered the clusters and classes so that the confusion matrix has large numbers near the diagonal. The fact that the numbers near the diagonal are larger than those away from the diagonals implies that many of the clusters can be identified with NSF organizations. For example, the 'theory' and 'chemistry' clusters can be identified with MPS, the 'social' cluster with SBE, the 'protein' and 'species' clusters with BIO and the 'ocean' cluster can be identified with GEO. Interestingly, some of the clusters indicate the multidisciplinary nature of some NSF organizations. The 'conference' cluster is seen to have many awards from MPS, ENG, SBE and BIO but no particular organization is dominant. Also, organizations such as MPS are seen to have subcategories — 'mathematical theory', 'chemistry reactions' and 'quantum physics'. The numbers in the confusion matrix also indicate the inter-relationships between various NSF orgainzations. For example, the 'materials' cluster indicates closeness between MPS and ENG while the 'design' cluster shows the closeness between ENG and CSE.

The quality measures of purity and entropy for this clustering are given in the following table.

theory(7442)	theory(.339),mathematical(.295),equations(.25),geometry(.192), problems(.186),sciences(.177),differential(.168) algebraic(.165),manifolds(.126),spaces(.114)
chemistry(7420)	chemistry(.448),reactions(.217),organic(.21),nmr(.194), molecules(.182),metal(.169),chemical(.151),compounds(.142), spectroscopy(.122),reaction(.12)
physics (9835)	physics(.189),quantum(.166),magnetic(.166),particle(.144), electron(.135),stars(.13),solar(.129),energy(.123), laser(.115),theoretical(.112)
materials(11589)	materials(.292),phase(.192),properties(.156),films(.14), polymer(.135),surface(.125),thin(.116),optical(.114), temperature(.11),liquid(.107)
design(13255)	design(.243),algorithms(.216),control(.18),parallel(.175), performance(.149),problems(.144),software(.118),models(.115), computer(.112),optimization(.1)
conference(6652)	conference(.47),workshop(.399),international(.23),held(.207), meeting(.181),symposium(.178),travel(.135),participants(.112), scientists(.107),researchers(.096)
mathematics(7374)	mathematics(.342),teachers(.333),school(.259),education(.177), college(.151),year(.150),teacher(.132),summer(.125), schools(.12)
laboratory(11302)	laboratory(.303),equipment(.241),undergraduate(.225), computer(.186),engineering(.185),courses(.175),student(.132), biology(.126),faculty(.124),projects(.124)
social(8267)	social(.27),economic(.180),political(.167),policy(.143), dissertation(.13),language(.126),public(.104),cultural(.103), decision(.1),market(.087)
protein(9212)	protein(.273),cell(.258),cells(.235),proteins(.22),gene(.196), genes(.179),dna(.153),expression(.138),molecular(.131), regulation(.119)
species(8822)	species(.326),plant(.167),populations(.151),plants(.133), population(.125),evolutionary(.125),evolution(.101), variation(.098),ecological(.096)
ocean(12546)	ocean(.232),ice(194),climate(.153),mantle(.147),sea(.134), water(.118),seismic(.106),circulation(.097),pacific(.096), isotopic(.092)

Table 2. Top 10 words associated with each cluster (the numbers in the right column give the word's weight in the concept vector)

Despite the above quality measures, it is sometimes best to have an informal way of examining a clustering. For this purpose, we have created a sequence of web pages to browse through the clusters, and the documents and keywords contained in them. The clustering presented above is available at http://www.cs.utexas.edu/users/dml/demos/nsfbrowser/.

	MPS	ENG	CSE	EHR	SBE	OPP	BIO	GEO
theory(7442)	**6502**	196	240	92	336	0	43	33
chemistry(7420)	**4747**	572	8	991	363	24	377	338
physics(9835)	**4901**	1649	31	146	849	147	176	1936
materials(11589)	3257	**7094**	33	171	541	12	89	392
design(13255)	952	4754	**5850**	357	840	5	349	148
conference(6652)	1149	**1767**	544	383	1176	146	947	540
mathematics(7374)	410	208	772	**5658**	143	21	65	97
laboratory(11302)	767	1344	541	**5658**	209	410	1036	1337
social(8267)	84	438	221	317	**6686**	126	326	69
protein(9212)	164	784	20	162	383	25	**7578**	96
species(8822)	61	153	7	121	1690	252	**5602**	936
ocean(12546)	43	406	8	56	509	1599	180	**9745**

Table 3. Confusion matrix between the 8 "true" classes and 12 clusters for the entire NSF collection

Cluster	Purity	Entropy
theory	.87369	.275307
chemistry	.639757	.585657
physics	.498322	.670075
materials	.612132	.499253
design	.441343	.643922
conference	.265634	.911829
mathematics	.76729	.429274
laboratory	.500619	.766223
social	.808758	.397713
protein	.822623	.347316
species	.635003	.535591
ocean	.776742	.389061

Table 4. Purity and entropy results for the computed clusters of the entire NSF collection

6. Conclusions and Future Work

In this paper, we have presented a time and memory efficient scheme for clustering very large document collections. We employ a multithreaded approach to reading and preprocessing the documents to mitigate disk I/O costs, and use the effective spherical k-means algorithm to cluster the documents. Our experimental results show that we are able to preprocess and cluster 113,716 NSF award abstracts in 23 minutes on a Sun workstation with modest memory consumption. We have also demonstrated that the quality of the clustering is good. Based on the experiments we have done in this paper, we predict that we can use our scheme to cluster a million documents into 12 clusters in less than 4 hours on a Sun workstation.

In future work, we will continue our focus on improving the efficiency and scalability of our scheme. The dot product computation between all the document vectors and concept vectors is the computational bottleneck in the spherical k-means algorithm. To cut down on this computation, techniques like "truncation" which project each document vector onto a small subspace of the total word space will be investigated, see [SS97]. Meanwhile, more sophisticated handling of I/O threads will be studied in order to cut down the I/O cost which is the bottleneck for preprocessing. Parallelizing the whole process can be one of the approaches. Hierarchical clustering will also be studied in future work to discover the inherent hierarchical structure and the 'correct' number of clusters in the data set.

References

[BGG$^+$98] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 1998.

[Cal99] Brent Callaghan. *NFS Illustrated.* Addison-Wesley, 1999.

[CKPT92] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR*, 1992.

[DGL89] I. Duff, R. Grimes, and J. Lewis. Sparse matrix test problems. *ACM Trans Math Soft*, pages 1–14, 1989.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[DM01] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001. Also appears as IBM Research Report RJ 10147, July 1999.

[FBY92] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[GJW82] M. R. Garey, D. S. Johnson, and H. S. Witsenhausen. The complexity of the generalized Lloyd-Max problem. *IEEE Trans. Inform. Theory*, 28(2):255–256, 1982.

[Hea78] J. Heaps. *Information Retrieval - Computational and Theoretical Aspects.* Academic Press, 1978.

[Kol97] T. G. Kolda. *Limited-Memory Matrix Methods with Applications.* PhD thesis, The Applied Mathematics Program, University of Maryland, College Park, Mayland, 1997.

[KPR98] Jon Kleinberg, C. H. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. *Data Mining and Knowledge Discovery*, 2(4):311–324, December 1998.

[Mit97] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[MS96] D. Musser and A. Saini. *STL Tutorial and Reference Guide.* Addison-Wesley, 1996.

[NBF96] Bradford Nichols, Bick Buttlar, and Jackie Proulx Farrell. *Pthreads Programming.* O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1996.

[Pax96] Vern Paxson. Flex user manual, November 1996.

[Ras92] E. Rasmussen. Clustering algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[SB88] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):513–523, 1988.

[SGM00] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the AAAI2000 Workshop on Artificial Intelligence for Web Search*, pages 58–64, Austin, Texas, July 2000. AAAI/MIT Press.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Retrieval.* McGraw-Hill Book Company, 1983.

[SS97] H. Schütze and C. Silverstein. Projections for efficient document clustering. In *ACM SIGIR*, 1997.

[Wil88] P. Willet. Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597, 1988.

[ZE98] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *ACM SIGIR*, 1998.

[Zip49] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison Wesley, Reading, MA, 1949.

Chapter 21

A SCALABLE HIERARCHICAL ALGORITHM FOR UNSUPERVISED CLUSTERING

Daniel Boley

Abstract Top-down hierarchical clustering can be done in a scalable way. Here we describe a scalable unsupervised clustering algorithm designed for large datasets from a variety of applications. The method constructs a tree of nested clusters top-down, where each cluster in the tree is split according to the leading principal direction. We use a fast principal direction solver to achieve a fast overall method. The algorithm can be applied to any dataset whose entries can be embedded in a high dimensional Euclidean space, and takes full advantage of any sparsity present in the data. We show the performance of the method on text document data, in terms of both scalability and quality of clusters. We demonstrate the versatility of the method in different domains by showing results from text documents, human cancer gene expression data, and astrophysical data. For that last domain, we use an out of core variant of the underlying method which is capable of efficiently clustering large datasets using only a relatively small memory partition.

Keywords: Unsupervised Clustering, hierarchical clustering, text mining, genomics, sparse matrices, principal directions

1. Introduction

Explosive growth in the volume of data available electronically has created a need to be able to automatically explore large data collections. Unsupervised clustering algorithms are classical tools which have increasingly been reexamined for their applicability to data mining efforts.

Ideally, these algorithms would be fast and scalable, require little or no a-priori understanding of the data contents or attributes and need no costly graph building or association rule preprocessing. In many applications it would be useful if the algorithm could also impose a natural hierarchy on the data set, compute properties for the set as a whole, handle cases where attribute information is missing, and be independent of the order in which the data is presented. Principal Direction Divisive Partitioning (PDDP) is such an algorithm [Bol98].

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 383–400.
© 2001 *Kluwer Academic Publishers.*

Originally applied in the context of text documents retrieved from the WWW as part of the WebACE project [BGG+99b], PDDP has proven to be computationally efficient while providing high quality clusters. In addition to producing a partitioning of the data, the PDDP method

- yields weights showing which attributes were most significant in distinguishing the contents of one cluster from another,
- implements an automated stopping test based on the distribution of the data,
- allows a straightforward way to process datasets with missing attribute values,
- generates a hierarchical tree of clusters which can easily be updated locally, and
- is independent of any particular ordering of the input data, without using any randomized starting conditions.

The method has been successfully applied in a variety of application domains in addition to text documents, such as vision-based texture analysis and movie recommendation services.

2. The PDDP Algorithm

The PDDP algorithm employs the vector space model, where each data sample is represented by a vector of numerical attribute values. The data samples are embedded in a very high dimension Euclidean space, and the algorithm partitions this space with a collection of hyperplanes calculated to achieve good separation among the data samples. The data space is separated by a hyperplane into two half-spaces. The process continues recursively by separating each half-space with new hyperplanes computed independently. The method builds a binary tree of many polytope regions from the top down, until a stopping test is satisfied.

Since the PDDP algorithm operates directly with the collection of numerical attribute vectors, only a limited amount of preprocessing is necessary to generate the input data necessary for PDDP. This method was originally developed as part of the WebACE Project [BGG+99b] in the context of text documents where each document is represented by a scaled vector of word counts. The preprocessing consisted of removing the stop words and common word endings, and counting the number of occurrences of each word in each document. The result was an n-vector $\mathbf{d}$ of word counts associated with each document. All these vectors were combined into a single $n \times m$ matrix $\mathbf{M}$ in which each column corresponded to a document and each row corresponded to a particular word. In this domain,

```
Algorithm PDDP.
0. Start with $n \times m$ matrix $\mathbf{M}$ of vectors, one for each
       data sample, and a desired number of clusters $c_{\max}$.
1. Initialize Binary Tree with a single Root Node.
2. For $c = 2, 3, \ldots, c_{\max}$ do
3.     Select leaf node C with largest ScatterValue (21.2),
           and L & R := left & right children of C [step (a) in the text].
4.     Compute $\mathbf{v}_{\mathrm{C}} = g_{\mathrm{C}}(\mathbf{M}_{\mathrm{C}}) \equiv \mathbf{u}_{\mathrm{C}}^T(\mathbf{M}_{\mathrm{C}} - \mathbf{w}_{\mathrm{C}}\mathbf{e}^T)$
5.     For $i \in$ C, if $v_i \leq 0$, assign data sample $i$ to L,
                 else     assign it to R [step (b) in the text].
6. Result: A binary tree with $c_{\max}$ leaf nodes
       forming a partitioning of the entire data set.
```

Figure 1. Summary of the method to do a full build of the PDDP tree from scratch. Here $\mathbf{M}_{\mathrm{C}}$ is the matrix of data vectors for the data samples in cluster C, and $\mathbf{w}_{\mathrm{C}}, \mathbf{u}_{\mathrm{C}}$ are the centroid and principal direction vectors, respectively for C.

the matrix was generally very sparse, often less than 1% of the entries were nonzero. This sparsity results in a very fast and memory efficient method for carrying out the splitting process. However, this algorithm is not restricted to text domains and here we describe it in general terms.

2.1. Basic Algorithm Description

In the general situation, each data sample is represented by an n-vector of attribute values, and all these vectors (treated as column vectors) are assembled into an $n \times m$ data matrix $\mathbf{M}$. The clustering via PDDP is a recursive process that operates directly on the matrix $\mathbf{M}$. PDDP starts with a single "cluster" encompassing the entire dataset, and then divides this cluster into subclusters recursively using a two step process. At each stage, PDDP (a) selects a cluster to split, and (b) splits that cluster into two subclusters which become children of the original cluster. The result is a binary tree hierarchy imposed on the data collection. At every stage, the leaf nodes in the tree form a partition of the entire data collection. In the process of going to the next stage, one of those leaf nodes is selected and split in two. The behavior of the algorithm is controlled by the methods used to accomplish steps (a) and (b), and these methods are independent of one another. For step (a), PDDP usually selects the cluster with the largest *scatter value* (defined in the next subsection), which is the sum of all the squared distances from each data vector to the cluster centroid $\mathbf{w}$, though any suitable criterion can be used.

Once selected in step (a), the node is split in step (b), and this splitting process is the single most expensive step in the whole computation. The key to the computational efficiency of the entire approach is the efficient computation of the vectors needed in this step. Suppose PDDP were to split cluster C consisting of k data samples of attribute values. It places each data sample $\mathbf{d}$ in the left or right child of cluster C according to the sign of the linear discriminant function

$$g_{\mathrm{C}}(\mathbf{d}) = \mathbf{u}_{\mathrm{C}}^T(\mathbf{d} - \mathbf{w}_{\mathrm{C}}) = \sum_{i \in \mathrm{C}} u_i(d_i - w_i), \tag{21.1}$$

where $\mathbf{u}_{\mathrm{C}}$, $\mathbf{w}_{\mathrm{C}}$ are vectors associated with C to be determined. If $g_{\mathrm{C}}(\mathbf{d}) \leq 0$, the data sample $\mathbf{d}$ is placed in the new left child, otherwise $\mathbf{d}$ is placed in the new right child. Thus the behavior of the algorithm at each node in the binary tree is determined entirely by the two vectors $\mathbf{u}_{\mathrm{C}}$, $\mathbf{w}_{\mathrm{C}}$ associated with the cluster C.

The vector $\mathbf{w}_{\mathrm{C}} \overset{\mathrm{def}}{=} (1/k) \sum_{j \in \mathrm{C}} \mathbf{d}_j$ is the *mean* or *centroid* vector. The vector $\mathbf{u}_{\mathrm{C}}$ is the direction of maximal variance, also known as the leading left singular vector for the matrix $\mathbf{M}_{\mathrm{C}} - \mathbf{w}_{\mathrm{C}}\mathbf{e}^T$. This direction corresponds to the largest eigenvalue of the sample covariance matrix for the cluster. Here $\mathbf{M}_{\mathrm{C}}$ is the matrix of columns of data samples in cluster C. The computation of $\mathbf{u}_{\mathrm{C}}$ is the most costly part of this step. It can be performed quickly using a Lanczos-based solver for the singular values of the data matrix. This algorithm is very efficient, especially since low accuracy is all that is required, and can take full advantage of any sparsity present in the data.

The overall method can be summarized in Figure 1. As the method is "divisive" in nature, splitting each cluster into exactly two pieces at each step, the result is a binary tree whose leaf nodes are the sought-after clusters.

We use the classical "iris" data collection (see [DH73, p218] and references therein) to give a simple description of the structures produced by the PDDP algorithm. This data collection consists of 150 flowers: numbers 1-50 are of type *setosa*, numbers 51-100 *versicolor*, and number 101-150 *virginica*. To illustrate the binary tree on a simple case, data from 6 flowers in the set were chosen : 1, 2, 51, 52, 101, 102 with attributes shown in Fig. 2.

Fig. 3 shows the binary tree that results when the PDDP algorithm is used to to split this collection of six flowers into 3 clusters. The top box in Fig. 3 represents the root: it contains the indices of all six flowers. The column headed *centroid* is the centroid vector for all six flowers, and the column headed *direction* is the principal direction vector for all six flowers. The root has two children, one of which is a leaf node. In the leaf nodes, the principal direction vectors are not computed because they are not needed. For the non-leaf nodes, shown are the centroid and principal direction vector for the four flowers 51, 52, 101, 102. In this simple case, the PDDP algorithm partitioned the flowers consistently with their types. At each stage, the scatter value was used to select the next node to split, and an automatic stopping test employed using this scatter value (see §2.2 below). Though the scatter values are not shown in Fig. 3, the interior non-leaf node had a higher scatter than its "sibling" leaf node.

flower	1	2	51	52	101	102
type	setosa	setosa	versicolor	versicolor	virginica	virginica
sepal length	5.10	4.90	7.00	6.40	6.30	5.80
sepal width	3.50	3.00	3.20	3.20	3.30	2.70
petal length	1.40	1.40	4.70	4.50	6.00	5.10
petal width	0.20	0.20	1.40	1.50	2.50	1.90

Figure 2. Raw attributes for six flowers selected to illustrate the PDDP algorithm in Fig. 3.

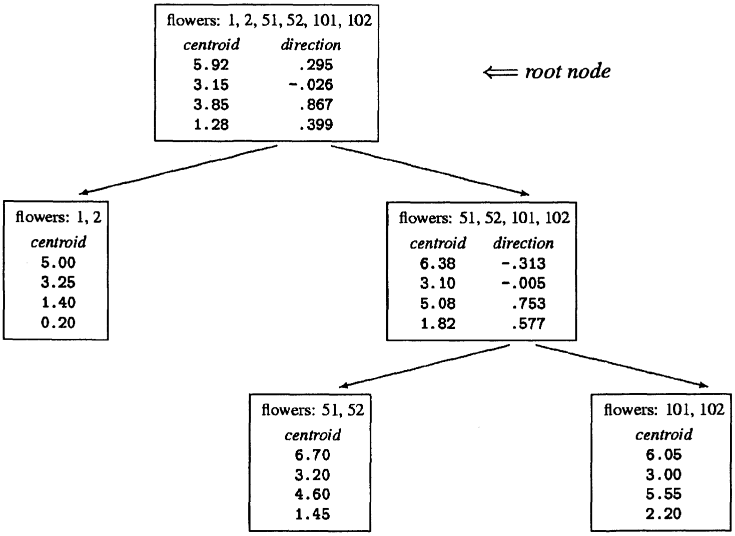

flowers: 1, 2, 51, 52, 101, 102
centroid
direction
5.92 .295
3.15 -.026
3.85 .867
1.28 .399
⟸ root node
flowers: 1, 2
centroid
5.00
3.25
1.40
0.20
flowers: 51, 52, 101, 102
centroid
direction
6.38 -.313
3.10 -.005
5.08 .753
1.82 .577
flowers: 51, 52
centroid
6.70
3.20
4.60
1.45
flowers: 101, 102
centroid
6.05
3.00
5.55
2.20

Figure 3. PDDP binary tree generated from the six irises shown in Fig. 2. Each box represents a node, listing the indices of the flowers represented by that node together with the average values of the attributes over all flowers in that node ("*centroid vector*"). In the two non-leaf nodes, the "*principal direction vector*" is shown, which is not computed at all for the leaf nodes.

2.2. Stopping Test

To make a working implementation, it is necessary to choose a stopping test, and in many domains, also a way to handle missing data.

Unless there is some other underlying reason to choose a particular number of clusters, the following stopping test can be used. It has been very successful on text documents, giving a number of clusters approximately equal to that computed by other methods tried. There are two components to the general stopping test, (a) a measure of the scatter for each individual cluster and (b) a measure of the relative separation between the clusters. In a top-down algorithm, the former should decrease while the latter should increase. Hence these values lead to a stopping test based on the ratio between these two values. Specifically, the method stops when the ratio of the individual cluster scatter

(a) to the cluster separation measure (b) decreases below a given threshold. It only remains to determine how to compute these two quantities.

For cluster scatter (a), we use a quantity already computed to decide which cluster should be split at each stage (step 3 in Fig. 1). This scatter value is defined by sum of squares of the distances from the individual data sample within a cluster to the cluster centroid:

$$\texttt{ScatterValue}_{\mathrm{C}} \stackrel{\text{def}}{=} \sum_{j \in \mathrm{C}} (\mathbf{d}_j - \mathbf{w}_{\mathrm{C}})^2 = \|\mathbf{M}_{\mathrm{C}} - \mathbf{w}_{\mathrm{C}}\mathbf{e}^T\|_F^2, \tag{21.2}$$

which can be efficiently computed as the Frobenius norm of the cluster matrix.

For the cluster separation measure (b), we use an approximation which can be computed efficiently, even if it is less effective than other measures. The approximation is computed by collecting the centroid vectors for all the individual clusters at each stage and computing their mutual scatter value. That is, let $\mathbf{W} \stackrel{\text{def}}{=} (\,\mathbf{w}_{\mathrm{C}_1} \ \cdots \ \mathbf{w}_{\mathrm{C}_c}\,)$ be the $n \times c$ matrix of all the collected centers of the c individual clusters existing at stage c. Then their mutual scatter value is

$$\texttt{CentroidScatter} \stackrel{\text{def}}{=} \sum_{j=1}^{c} (\mathbf{w}_{\mathrm{C}_j} - \tilde{\mathbf{w}})^2 = \|\mathbf{W} - \tilde{\mathbf{w}}\mathbf{e}^T\|_F^2, \tag{21.3}$$

where $\tilde{\mathbf{w}} = \frac{1}{c}\mathbf{W}\mathbf{e}$ is the centroid of the collected centroids.

Then the stopping test adopted for the PDDP method is the ratio

$$\frac{\max_{j=1}^{c} \texttt{ScatterValue}_{\mathrm{C}_j}}{\texttt{CentroidScatter}} \le \textit{some fixed threshold value}. \tag{21.4}$$

The *threshold value* is usually set to 1, but it can be set to larger or smaller values to obtain coarser or finer clusters, respectively.

2.3. Missing Values

Missing values often appear in datasets, representing data that is unknown as opposed to "zero." It is desirable to avoid having these values affect the placement of data samples into clusters. In the presence of missing data, it is a simple matter to compute the centroid vector $\mathbf{w}_{\mathrm{C}}$ by averaging each attribute value only over the non-missing values. Once the centroid vector is computed, we are able to replace each missing value with the corresponding average value for that attribute. With this replacement, the corresponding entries in $\mathbf{M}_{\mathrm{C}} - \mathbf{w}_{\mathrm{C}}\mathbf{e}^T$ are zero and hence have no contribution to the principal direction $\mathbf{u}_{\mathrm{C}}$ or the linear discriminant value (21.1). This replacement is temporary, and once the splitting process is completed each missing entry is reset to its original 'missing' value before proceeding to the next split. In this fashion missing values never push samples into the left or right child of a cluster, but the choice of which child cluster receives a given sample is based only on the non-missing attribute values.

cluster:	1	2	3
setosa	50	0	0
versicolor	0	46	4
virginica	0	0	50

Figure 4. PDDP Clustering result (confusion matrix) on the entire dataset of 150 irises, using norm scaling on the input data and the scatter-based automated stopping test (21.4) [with a threshold value of 2]. Each entry in the confusion matrix is a count showing the number of flowers of that type in the computed PDDP cluster. When the threshold value is set to 1, cluster 3 is split in two.

3. Performance

Performance testing has demonstrated PDDP provides high quality clusters at a relatively low computational cost. Most of the experience has been on text documents, but the algorithm has also been successful on datasets of movie ratings, texture images, toxicity databases, etc. In this section we summarize the main performance results from the experience on text documents. For example, using the PDDP algorithm with the stopping test on the entire "iris" dataset, the algorithm yielded the clusters shown in Fig. 4.

3.1. Speed

The cost of the PDDP method depends almost entirely on the cost of its most expensive step, which is the computation of the principal direction vector. The computation of this vector is carried out with a Lanczos-based eigensolver [GV96], whose cost is proportional to the number of nonzeroes in the data matrix. Thus the PDDP method scales linearly with the size of the data matrix. This behavior is shown in Fig. 5, where it is seen that the cost depends more on the number of nonzeroes (the horizontal axis) than the actual number of documents or words. For example, Fig. 5 shows that the time to obtain the clusters shown in Fig. 6 was approximately 37 seconds on an SGI Challenge workstation.

To compare the cost of PDDP with that of more classical methods, we applied PDDP, Hypergraph [HKKM98], K-means - LSI [BDO95], Agglomeration [DH73], and AutoClass [CS96] methods to a text document collection of 185 documents with 10538 word dictionary. The first three methods all took under 2 minutes a 185MHz Sun workstation, but Agglomeration and Autoclass each took at least 30 minutes. It is difficult to compare PDDP with the more classical methods on large examples because, unlike PDDP, most classical methods do not scale linearly with the size of the problem. In our experiments on larger datasets, unmodified Agglomeration and Autoclass became prohibitively expensive. We remark that Agglomeration is a classical algorithm which generates a tree bottom-up that is very similar to that produced top-down by PDDP.

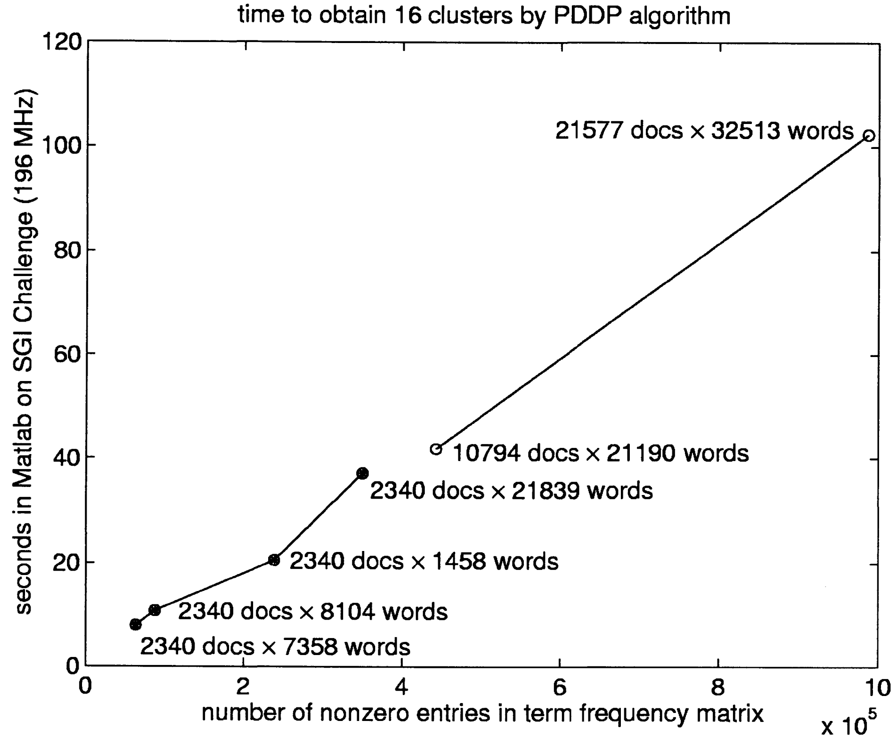


Figure 5. Time to compute 16 clusters for various datasets using interpreted MATLAB code on an SGI challenge workstation (196 MHz), against the number of nonzeroes in the data matrix.

Agglomeration is well known to be very effective, but it depends on the choice of "inter-cluster" separation, and it begins by computing the separation between every pair of data samples (documents), treating them as "singlet clusters." This is an $O(m^2)$ process, where m is the number of data samples. By randomly splitting the samples into $\sqrt{m}$ initial clusters, applying Agglomeration to each cluster, and then agglomerating the resulting clusters, one can reduce the total asymptotic running time to $O(m)$ (see e.g. [CKPT92]), but the results depend on the random initial splitting which leads to variability in the final clustering.

Other algorithms also exist which can also lead to high quality results (e.g. k–nearest neighbor or k–means) [GJJ96], but most suffer from high cost (e.g. nearest neighbor) or depend critically on a good starting point which is often chosen randomly (e.g. k-means). BIRCH [ZRL96] incrementally builds a tree during a single pass over the data, and sweeping over the data in a different order could change the result.

cluster:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
business	90	0	0	0	7	0	5	12	0	6	0	1	18	3	0	0
health	0	150	166	171	3	0	1	1	0	0	0	0	0	2	0	0
politics	2	0	0	0	100	1	2	0	0	1	0	2	1	5	0	0
sports	0	0	0	0	1	62	35	0	0	1	0	0	0	42	0	0
technology	8	0	0	0	0	1	14	24	0	8	0	1	4	0	0	0
entertain.	24	0	0	4	11	4	22	61	135	131	148	159	143	137	204	206

Figure 6. Confusion Matrix showing how documents were distributed to 16 different clusters by topic labels. This data set has 2340 documents, and was based on a dictionary of 21839 words. Further subdivisions are necessary to classify the large collection of entertainment documents in this set.

method	*entropy*
Agglomeration - norm scaling [DH73]	0.684
PDDP - norm scaling	0.689
Hypergraph [HKKM98]	0.787
K-means - LSI [BDO95]	0.837
PDDP - TFIDF scaling [SB88]	1.057
AutoClass [CS96]	2.049
Agglomeration - TFIDF scaling	2.339

Figure 7. Entropies by various methods on a set of 185 documents with a 10538 word dictionary.

3.2. Quality of Clusters

There is no absolute scale to measure the quality of clusters, but one can see the so-called *confusion matrix* for a dataset consisting of 2,340 text documents using a dictionary of 21839 words in Fig. 6. The confusion matrix shows how the different types of documents in six broad categories were distributed among 16 clusters.

To be able to compare the quality of the clusters produced by PDDP with that arising from other methods, it is necessary to quantify the measure of quality using, for example, an entropy measure as in [Bol98, BGG$^+$99a]. This entropy measure requires that the documents be given category labels in advance by hand, and hence it is difficult to apply this measure to very large datasets. However, for the dataset of 185 documents used to compare the costs of PDDP with other methods mentioned above, it was relatively easy to assign topic labels to the documents. These labels were not used by any of the classification methods discussed in this paper or for any of the performances tests. The labels were used only to measure the entropy of the resulting clusters as a measure of quality. The entropy of a given cluster $\mathtt{C}$ is defined by

$$e_{\mathtt{C}} = -\sum_i \left(\frac{c(i,\mathtt{C})}{\sum_i c(i,\mathtt{C})}\right) \cdot \log\left(\frac{c(i,\mathtt{C})}{\sum_i c(i,\mathtt{C})}\right),$$

where $c(i, \mathtt{C})$ is the number of times label i occurs in cluster $\mathtt{C}$. A cluster has zero entropy if the all the documents in the cluster have the same label, otherwise it has a positive entropy. The total entropy is the weighted average of the individual cluster entropies:

$$e_{\text{total}} = \frac{1}{m} \sum_{\mathtt{C}} e_{\mathtt{C}} \cdot [\textit{number of documents in cluster } \mathtt{C}].$$

As a consequence, the lower the entropy the better the quality.

Various clustering methods were applied to this dataset, and the results are shown in Fig. 7. Each method was forced to produce the same number of clusters in order to make the results consistent.

4. Scientific Data Set Examples

We illustrate how this clustering method performs on scientific data sets by using two examples, one drawn from gene expression data and one drawn from an astronomical sky survey. The aim is to show the kinds of results that can be obtained in an automated way from the PDDP clustering algorithm. The goal is to allow easier exploration of large datasets, facilitating the extraction of new patterns and novel discoveries from the data. The description of novel discoveries themselves is beyond the scope of this paper.

4.1. Gene Expression Data

Recent years has seen the explosion of genetic data available electronically. Much genetic data is structural, such as the genetic sequences of base pairs making up the DNA in living cells. Recently, however, there has been much interest in functional genetic data, such as showing which genes are active (i.e. *expressed*) under a variety of external conditions. Such information can be used to discover the purpose of many genes as well as to identify which genes are prone to "misbehavior," thereby causing disease. For example, it is well known that cancer is often caused by genes behaving improperly giving rise to uncontrolled growth in cells. It has been found recently that some cancers have a distinctive gene signature which can be used to both refine diagnoses and to focus treatments. We use the data from one such study on the gene signature for a certain class of lymphomas [AED$^+$00] to illustrate the kinds of analysis that the PDDP produces in an automated fashion.

The gene expression data in [AED$^+$00] consists of microarray assays of 4026 genes on 96 different tissue samples obtained from patients with a certain class of non-Hodgkins lymphoma. The goal was to identify which genes were important for different kinds of cancer, both for the purpose of distinguishing between cancers and for the purpose of designing therapies tailored for each particular cancer. The data consists of a 4026 × 96 matrix of attribute values

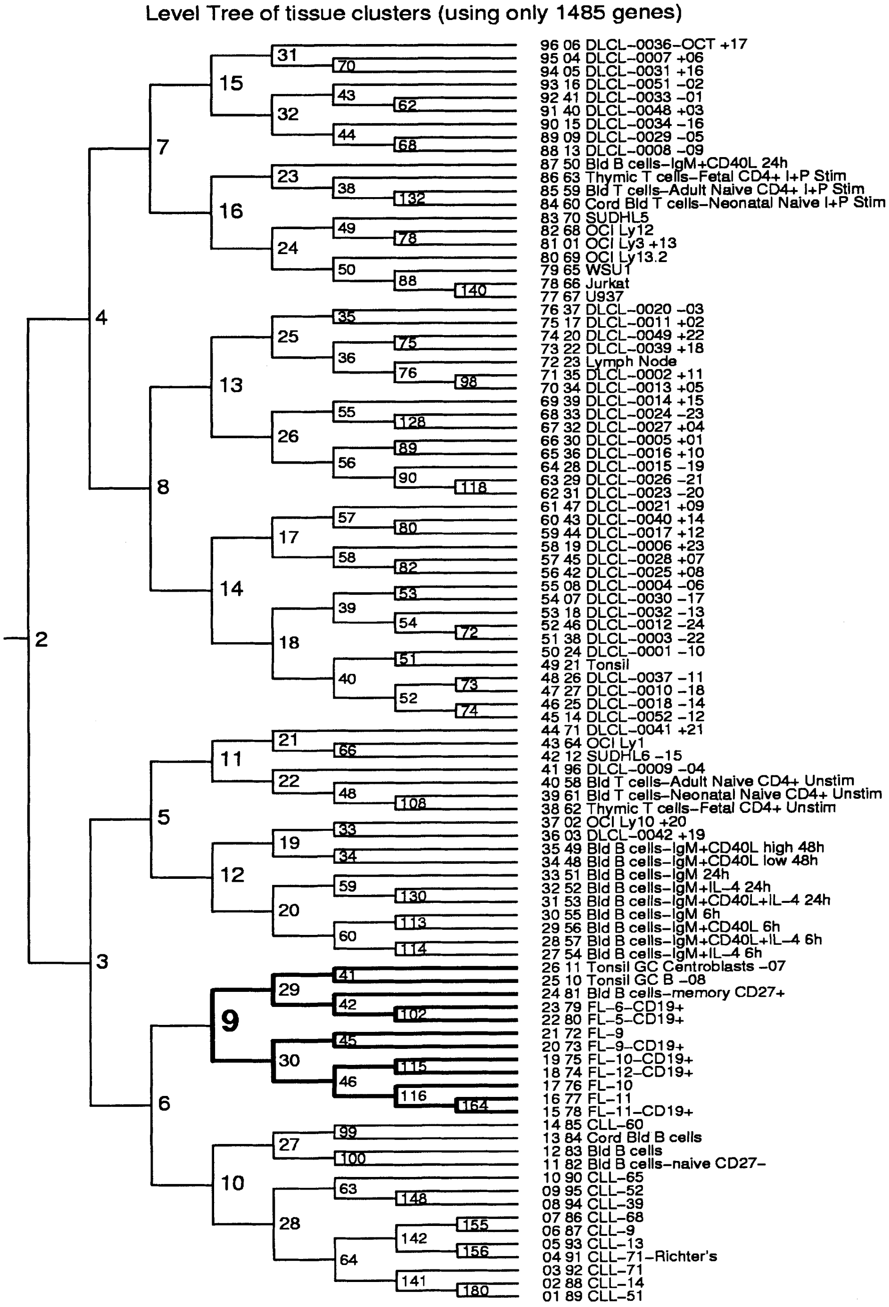
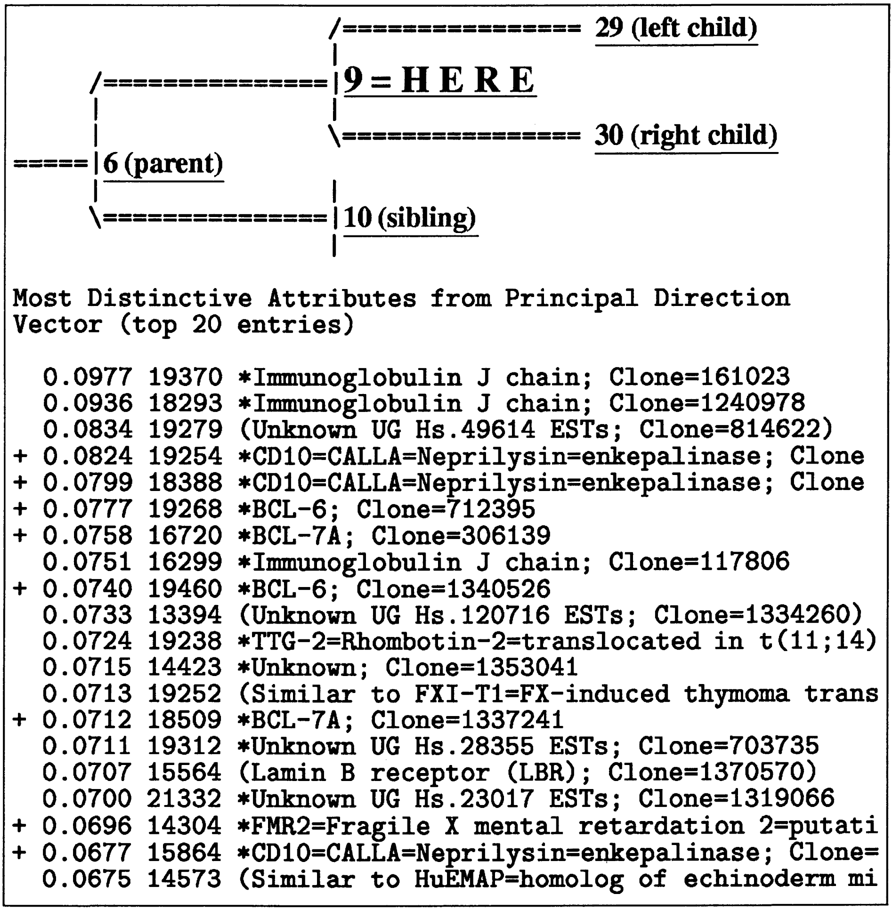

Level Tree of tissue clusters (using only 1485 genes)
31
70
15
43
62
32
44
68
7
23
38
132
16
49
78
24
50
88
140
4
35
25
75
36
76
98
13
55
128
26
89
56
90
118
8
57
17
80
58
82
14
53
39
54
72
2
18
51
40
73
52
74
21
66
11
22
48
108
5
33
19
34
12
59
130
20
113
60
114
3
41
29
42
102
9
45
30
115
46
116
164
6
99
27
100
10
63
148
28
155
142
156
64
141
180
96 06 DLCL-0036-OCT +17
95 04 DLCL-0007 +06
94 05 DLCL-0031 +16
93 16 DLCL-0051 -02
92 41 DLCL-0033 -01
91 40 DLCL-0048 +03
90 15 DLCL-0034 -16
89 09 DLCL-0029 -05
88 13 DLCL-0008 -09
87 50 Bld B cells-IgM+CD40L 24h
86 63 Thymic T cells-Fetal CD4+ I+P Stim
85 59 Bld T cells-Adult Naive CD4+ I+P Stim
84 60 Cord Bld T cells-Neonatal Naive I+P Stim
83 70 SUDHL5
82 68 OCI Ly12
81 01 OCI Ly3 +13
80 69 OCI Ly13.2
79 65 WSU1
78 66 Jurkat
77 67 U937
76 37 DLCL-0020 -03
75 17 DLCL-0011 +02
74 20 DLCL-0049 +22
73 22 DLCL-0039 +18
72 23 Lymph Node
71 35 DLCL-0002 +11
70 34 DLCL-0013 +05
69 39 DLCL-0014 +15
68 33 DLCL-0024 -23
67 32 DLCL-0027 +04
66 30 DLCL-0005 +01
65 36 DLCL-0016 +10
64 28 DLCL-0015 -19
63 29 DLCL-0026 -21
62 31 DLCL-0023 -20
61 47 DLCL-0021 +09
60 43 DLCL-0040 +14
59 44 DLCL-0017 +12
58 19 DLCL-0006 +23
57 45 DLCL-0028 +07
56 42 DLCL-0025 +08
55 08 DLCL-0004 -06
54 07 DLCL-0030 -17
53 18 DLCL-0032 -13
52 46 DLCL-0012 -24
51 38 DLCL-0003 -22
50 24 DLCL-0001 -10
49 21 Tonsil
48 26 DLCL-0037 -11
47 27 DLCL-0010 -18
46 25 DLCL-0018 -14
45 14 DLCL-0052 -12
44 71 DLCL-0041 +21
43 64 OCI Ly1
42 12 SUDHL6 -15
41 96 DLCL-0009 -04
40 58 Bld T cells-Adult Naive CD4+ Unstim
39 61 Bld T cells-Neonatal Naive CD4+ Unstim
38 62 Thymic T cells-Fetal CD4+ Unstim
37 02 OCI Ly10 +20
36 03 DLCL-0042 +19
35 49 Bld B cells-IgM+CD40L high 48h
34 48 Bld B cells-IgM+CD40L low 48h
33 51 Bld B cells-IgM 24h
32 52 Bld B cells-IgM+IL-4 24h
31 53 Bld B cells-IgM+CD40L+IL-4 24h
30 55 Bld B cells-IgM 6h
29 56 Bld B cells-IgM+CD40L 6h
28 57 Bld B cells-IgM+CD40L+IL-4 6h
27 54 Bld B cells-IgM+IL-4 6h
26 11 Tonsil GC Centroblasts -07
25 10 Tonsil GC B -08
24 81 Bld B cells-memory CD27+
23 79 FL-6-CD19+
22 80 FL-5-CD19+
21 72 FL-9
20 73 FL-9-CD19+
19 75 FL-10-CD19+
18 74 FL-12-CD19+
17 76 FL-10
16 77 FL-11
15 78 FL-11-CD19+
14 85 CLL-60
13 84 Cord Bld B cells
12 83 Bld B cells
11 82 Bld B cells-naive CD27-
10 90 CLL-65
09 95 CLL-52
08 94 CLL-39
07 86 CLL-68
06 87 CLL-9
05 93 CLL-13
04 91 CLL-71-Richter's
03 92 CLL-71
02 88 CLL-14
01 89 CLL-51

Figure 8. PDDP tree using the 1485 most distinctive genes from the 4026 × 96 gene expression matrix from [AED$^+$00]. Node 9 is illustrated in Fig. 9.



```
                    /================ 29 (left child)
                    |
     /==============|9 = H E R E
     |              |
     |              \================ 30 (right child)
=====|6 (parent)
     |              |
     \==============|10 (sibling)
                    |

Most Distinctive Attributes from Principal Direction
Vector (top 20 entries)

  0.0977 19370 *Immunoglobulin J chain; Clone=161023
  0.0936 18293 *Immunoglobulin J chain; Clone=1240978
  0.0834 19279 (Unknown UG Hs.49614 ESTs; Clone=814622)
+ 0.0824 19254 *CD10=CALLA=Neprilysin=enkepalinase; Clone
+ 0.0799 18388 *CD10=CALLA=Neprilysin=enkepalinase; Clone
+ 0.0777 19268 *BCL-6; Clone=712395
+ 0.0758 16720 *BCL-7A; Clone=306139
  0.0751 16299 *Immunoglobulin J chain; Clone=117806
+ 0.0740 19460 *BCL-6; Clone=1340526
  0.0733 13394 (Unknown UG Hs.120716 ESTs; Clone=1334260)
  0.0724 19238 *TTG-2=Rhombotin-2=translocated in t(11;14)
  0.0715 14423 *Unknown; Clone=1353041
  0.0713 19252 (Similar to FXI-T1=FX-induced thymoma trans
+ 0.0712 18509 *BCL-7A; Clone=1337241
  0.0711 19312 *Unknown UG Hs.28355 ESTs; Clone=703735
  0.0707 15564 (Lamin B receptor (LBR); Clone=1370570)
  0.0700 21332 *Unknown UG Hs.23017 ESTs; Clone=1319066
+ 0.0696 14304 *FMR2=Fragile X mental retardation 2=putati
+ 0.0677 15864 *CD10=CALLA=Neprilysin=enkepalinase; Clone=
  0.0675 14573 (Similar to HuEMAP=homolog of echinoderm mi
```

Figure 9. Excerpt from web browser display of Node 9 in the tree of Fig. 8 showing the genes distinguishing this Node from its sibling node. The symbol "+" has been added to identify the germinal centre B cell genes, (known to differ in FL compared to CLL).

in the range $[-2, 2]$ representing logarithms of the activity of each gene within each tissue. Approximately 5% of the data is missing because of faults during the collection of the experimental data. The authors of [AED$^+$00] applied hierarchical agglomeration to obtain a tree structure on the 96 tissue samples, and then manually examined the clusters within that tree to identify significant genes. PDDP also produces a very similar tree, but top-down instead of bottom-up.

The PDDP tree can be used in at least two ways on this dataset. The simplest is to use the principal direction information to identify which genes are significant in splitting *anywhere* in the tree. One may remove any gene for which the corresponding entry (weight) in any principal direction vector is small in absolute value. Such genes are never significant in splitting a cluster into subclusters using the inner product (21.1). Therefore the genes that remain are more useful in distinguishing among the tissues samples. In this experiment, we kept the genes whose weight in any principal direction vector exceeds a given threshold value. This value was chosen by noting that if all the entries in a principal direction vector had equal weight, the weight would have to be $1/\sqrt{n}$ because the vector is normalized to make the sum of squares equal to 1. To extract the significant genes, a value of 3 times this neutral value was chosen, namely $3/\sqrt{n} = .078$. This reduced the number of genes from 4062 down to 1485, and PDDP was then applied to this reduced set (96 tissues by 1485 genes). The PDDP trees produced using all 4026 genes and with only 1485 genes were very similar, and we show the latter tree in Fig. 8.

To allow a user to explore the tree and identify these distinctive genes, the software converts the tree from the internal representation into an HTML document which can be browsed using any standard web browser. As an example, an excerpt from a web browser display is reproduced in Fig. 9, showing the distinctive genes for the cluster shown in bold in Fig. 8. Of course, there is no intention to claim that the genes listed in Fig. 9 are all significant for this cluster or any other cluster, but we do remark that many of the listed genes were specifically identified in [AED$^+$00] as germinal centre B cells, known to distinguish the FL tissue samples in this cluster from the CLL tissue samples in the sibling cluster.

4.2. Astronomical Sky Survey

The advent of high-powered computational facilities has given rise to automated surveys of the sky with a view of cataloging and/or classifying the enormous number of objects that can be observed from Earth. A further goal of such surveys is to allow automated or semi-automated exploration to permit one to find "interesting" objects or clusters of objects for further study. Automated clustering methods would permit the exploration of this enourmous dataset in many dimensions, allowing astronomers to identify distinctive objects and relationsips between objects. In [PM99], *kd*-trees were proposed as a way to speed up a k-means clustering of Sloan Digital Sky Survey (a more recent automated sky survey) data. Though computed using different algorithms, *kd*-trees can be thought of as a special case of PDDP trees in which each separating hyperplane is normal to a coordinate axis. Hence PDDP trees share many of the same favorable properties, including yielding good seeds for the k-means method. This is an area which requires further research.

Preliminary experiments were carried out with data derived from the Minnesota Automated Plate Scan (APS) [PHO$^+$93, Web98]. This is a project to

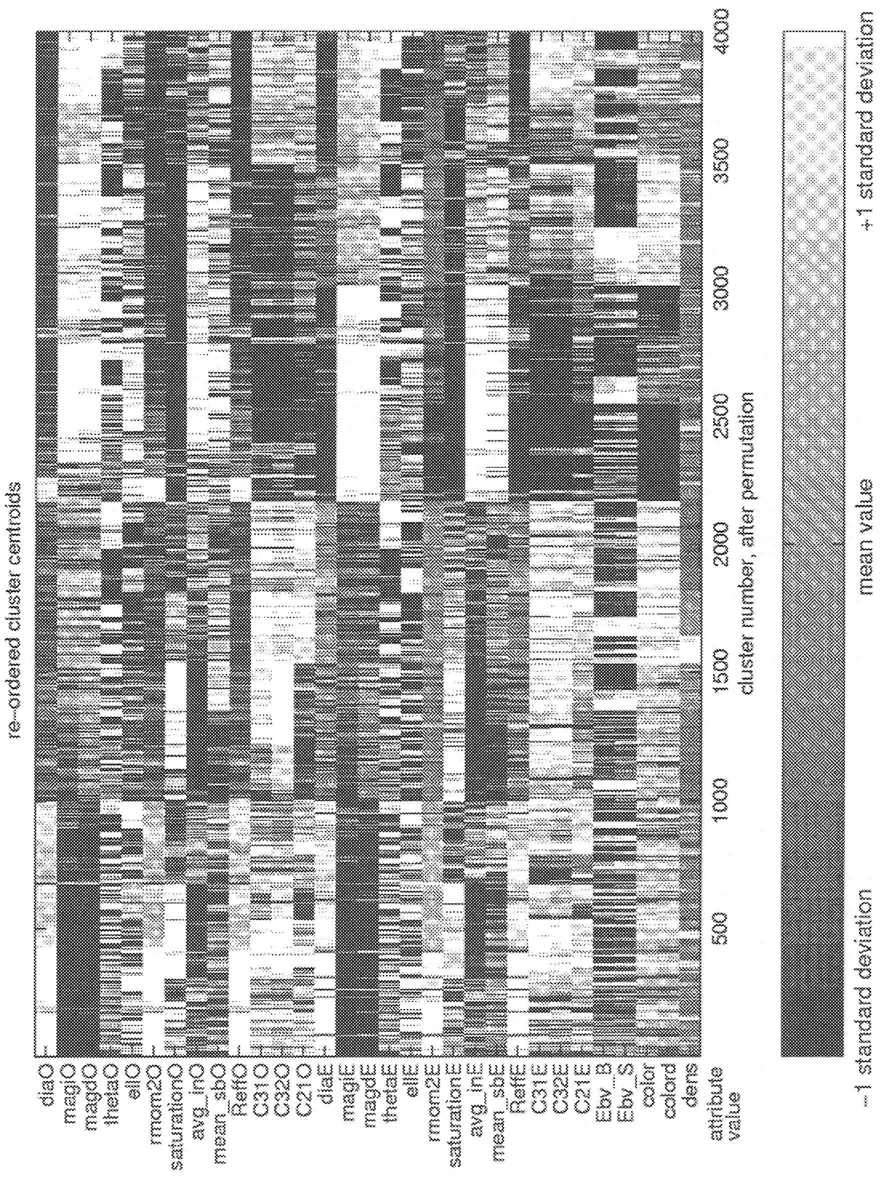


Figure 10. Individual centroid vectors for 4000 clusters computed from a sample APS dataset.

digitize the contents of photographic plates of the heavens from the Palomar Observatory Sky Survery (POSS I) originally produced in the 1950s, before the advent of artificial satellites.

The size of this data set spurred the development of an out-of-core version of the PDDP method, since the data set is too large to be loaded into memory at one time. This out-of-core version of the PDDP method has been designed for the situation where there is a large number of data samples, but relatively few attributes. Suppose the number of attributes n is much less than the number of data samples m, so that the data matrix $\mathbf{M}$ is very wide but not tall. In the following we use $\mathbf{e} \overset{\text{def}}{=} (1 \quad \cdots \quad 1)^T$ to denote a column vector of all ones of appropriate size. To compute the centroid vector $\mathbf{w}_{\mathrm{C}} = \frac{1}{k}\mathbf{M}_{\mathrm{C}}\mathbf{e}$ (where k is the size of the cluster C), it is necessary to make a pass over the entire matrix $\mathbf{M}_{\mathrm{C}}$. To compute the principal direction vector $\mathbf{u}_{\mathrm{C}}$, to the low accuracy needed for this algorithm, it suffices to accumulate the $n \times n$ matrix

$$\mathbf{A} \overset{\text{def}}{=} (\mathbf{M}_{\mathrm{C}} - \mathbf{w}_{\mathrm{C}}\mathbf{e}^T)(\mathbf{M}_{\mathrm{C}} - \mathbf{w}_{\mathrm{C}}\mathbf{e}^T)^T = \mathbf{M}_{\mathrm{C}}\mathbf{M}_{\mathrm{C}}^T - \mathbf{Z} - \mathbf{Z}^T + k\mathbf{w}_{\mathrm{C}}\mathbf{w}_{\mathrm{C}}^T,$$

where $\mathbf{Z} \overset{\text{def}}{=} \mathbf{M}_{\mathrm{C}}\mathbf{e}\mathbf{w}_{\mathrm{C}}^T = k\mathbf{w}_{\mathrm{C}}\mathbf{w}_{\mathrm{C}}^T$. Thus the matrix $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{M}_{\mathrm{C}}\mathbf{M}_{\mathrm{C}}^T - k\mathbf{w}_{\mathrm{C}}\mathbf{w}_{\mathrm{C}}^T$$

With one pass through the data, column by column, we can accumulate $\mathbf{M}_{\mathrm{C}}\mathbf{M}_{\mathrm{C}}^T = \sum_{j \in \mathrm{C}} \mathbf{d}_j\mathbf{d}_j^T$, and $\mathbf{w}_{\mathrm{C}} \overset{\text{def}}{=} (1/k)\sum_{j \in \mathrm{C}} \mathbf{d}_j$, and use the result to form $\mathbf{A}$ without any further access to the data. By saving the columns associated with each cluster in a separate scratch file as each cluster is formed, the passes over the data matrix can be made very fast.

As a small test case to demonstrate the out-of-core method, we used a small sample from the APS consisting of 212089 galactic objects, each with 26 attribute values. Using a 195 MHz SGI challenge machine (relatively old machine chosen to match that used to produce the timings in Fig. 5) and a Matlab implementation of the out-of-core algorithm, a PDDP tree with 4000 clusters can be computed in under 10 minutes. Of course it needs enough scratch disk space to store two copies of the data set, but the access to each copy of the data is sequential and hence reasonably efficient. Of course with enough memory and a compiled version, this can be computed much more efficiently with a standard in-core algorithm, but this experiment serves to demonstrate the practicality of the out-of-core algorithm. To visualize the results, we collected all the centroids for the 4000 clusters and displayed them side by side using color-coding to display the magnitude of each attribute value. The result is shown in Fig. 10, where it is seen that certain attributes appear to vary together over a portion of the data. The interpretation of these results is still an active area of research, especially an analysis of the utility of the encodings used in this digitization of the APS data set.

5. Conclusion

This brief introduction to the method of Principal Direction Divisive Partitioning is intended to show that this algorithm is a useful tool in situations which call for unsupervised clustering of large data collections which can be represented using very high dimensional numerical vectors. The PDDP method operates on the entire data collection as a unit, producing clusters of a quality comparable with those of the best alternative methods, but often at less cost. This brief description should be sufficient to show that the PDDP algorithm can be adapted to a wide variety of datasets and different situations, such as the presence of missing data, as well as providing useful information on the factors that were most significant in yielding the clusters that are eventually computed. The examples include some hints on the kinds of information which can be extracted from the PDDP tree, both to explain the results and to extract the most significant parts of the dataset.

Of course, there are limitations to the PDDP algorithm. The main limitations are PDDP's sensitivity to scaling of the data and the need to have all the data available at once. The sensitivity to scaling is a natural consequence of the way the data is represented with vectors of real numbers, and any algorithm using such a representation will suffer from the same limitation. The second limitation is the object of current research, where we have the goal of allowing the tree to be computed using a fraction of the data and updating the tree as new data becomes available.

Acknowledgments

This work was partially supported by NSF grant IIS-9811229.

References

[AED$^+$00] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson Jr, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

[BDO95] M. W. Berry, S. T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[BGG$^+$99a] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.

[BGG$^+$99b] D.L. Boley, M. Gini, R. Gross, E-H Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 13(5-6):365–391, 1999.

[Bol98] D. L. Boley. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[CKPT92] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *15th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*, pages 318–329, 1992.

[CS96] P. Cheeseman and J. Stutz. Bayesian classification (Autoclass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[GJJ96] Earl Gose, Richard Johnsonbaugh, and Steve Jost. *Pattern Recognition and Image Analysis*. Prentice Hall, 1996.

[GV96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 3rd edition, 1996.

[HKKM98] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *Bulletin of the Technical Committee on Data Engineering*, 21(1), 1998.

[PHO$^+$93] R.L. Pennington, R.M. Humphreys, S.C. Odewahn, W. Zumach, and P.M. Thurmes. The automated plate scanner catalog of the palomar sky survey – scanning parameters and procedures. *P. A. S. P.*, 105:521ff, 1993.

[PM99] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning conference on knowledge discovery in databases. In *Proc. of the Fifth Int'l Conference on Knowledge Discovery and Data Mining*, 1999.

[SB88] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[Web98] Web Site. The automated plate scanner, 1998. `http://aps.umn.edu`, supported by NASA.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Linvy. Birch: an efficient data clustering method for large databases. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, Montreal, Quebec, 1996.

Chapter 22

HIGH-PERFORMANCE SINGULAR VALUE DECOMPOSITION

David B. Skillicorn and Xiaolan Yang

Abstract Singular value decomposition is, among other things, a dimensionality reduction technique. It is used in data mining as a way to improve similarity measurements and as a preprocessing step before automatic clustering. We present several parallel algorithms for computing the SVD of a large matrix. Given a matrix with n rows and m columns, p-fold speedup of the computation part is achieved. Communication overheads range from $\mathcal{O}(m^2)$ to $\mathcal{O}(pm^2)$, which is smaller than the communication overheads of techniques based on parallelizing the inner loop of Hestenes-style algorithms.

Keywords: dimensionality reduction, SVD, incremental algorithms, parallel computing, cost modeling.

1. Motivation

Singular value decomposition (SVD) is a numerical technique that can be regarded as a space transformation. Suppose that we are given an n-by-m matrix. Each row can be regarded as a point in an m-dimensional space. SVD maps this m-dimensional space into a new m-dimensional space in which most of the variation among the original points is placed in the first dimension, the second largest amount of variation in the second dimension, and so on. Hence, each point can be thought of as being positioned in a k-dimensional space ($k \leq m$) simply by ignoring the last $m - k$ coordinates. Moreover, this k-dimensional space is the best approximation, in a strong sense, to the original space.

In data mining applications, the initial matrix is an array of objects and attributes. The number of rows, n, of the matrix is typically very large, in the range 10^3–10^9. The number of columns, m, is also large (although perhaps only in the range 10-10^4). However, this is large

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 401–424.
© 2001 *Kluwer Academic Publishers.*

enough for many of the difficulties of working in high dimension (the so-called "curse of dimensionality") to play a significant role.

Singular value decomposition is therefore useful in almost any high-dimensional setting as a tool for dimensionality reduction. It may be used for preprocessing before using an automatic clustering package such as Autoclass. It may be used to improve similarity measurements, such as those required in image retrieval [ZKea98] or collaborative filtering [Pry98].

The most successful application of SVD for reducing dimensionality is in the area of text retrieval, where it has become known as *latent semantic indexing* [DDF90, BF96]. In a document repository, a matrix is constructed whose columns are indexed by documents and whose rows are indexed by index terms. The matrix entry A_{ij} counts the frequency of occurrence of term i in document j. Vector-based text retrieval uses Euclidean distance within the space whose axes are documents as a similarity measure. However, the number of documents, and hence the number of dimensions of the vector space, is extremely large and so badly-behaved under distance metrics. The use of singular value decomposition to transform this vector space into one of much lower dimension produced a radical improvement in retrieval performance. Furthermore, it has the useful side effect of mapping index terms whose underlying meaning is the same to similar positions, based on their cognate use in different documents This process is revealing enough to permit querying in a language different from that of the documents themselves [BC97].

Because singular value decomposition is an expensive technique and is being applied to large matrices, it is natural to consider ways of improving its performance. In this paper we consider two such improvements. The first applies when the matrix itself changes over time – this is often the case when the data being modelled is sampled from continuously time-varying processes. In this case, we illustrate the use of an incremental algorithm.

The second is the use of parallelism to provide speedups in overall performance. We present a set of approximate algorithms and an exact algorithm that provide approximately linear speedup.

Section 2 introduces the singular value decomposition technique and some standard algorithms to compute it. Section 3 describes an incremental algorithm for computing SVD if the SVD of a similar matrix is already known. Section 4 introduces a parallelization approach aimed at producing large-grained parallel processes. Section 5 describes a cost model that is used to compare the performance of different parallel algorithms. Section 6 describes two approximate parallel algorithms and demonstrates the quality of their results. Section 7 describes a truly parallel algorithm. Section 8 summarizes the paper.

2. Introduction to SVD

Given an n-by-m matrix A with $n \geq m$, the singular value decomposition of A is

$$A = U\Sigma V^T$$

where U is n-by-m, Σ is m-by-m, and V is m-by-m. U and V are orthonormal, and Σ is a diagonal matrix whose elements are a set of non-negative singular values, σ_1, σ_2, $\ldots$, σ_r (where r is the rank of A, $r \leq m$).

The rows of U represent coordinates of the corresponding rows of A in a space whose axes are spanned by the columns of V. Symmetrically, the rows of V represent the coordinates of the corresponding columns of A in a space whose axes are given by the columns of U. The magnitudes of the corresponding singular values describe the amount of variation captured in each dimension.

The standard way to compute the SVD is the Golub-Reinsch algorithm [GvL96], which proceeds by two-sided reduction to bidiagonal form followed by the QR algorithm.

An alternative algorithm for computing the SVD is one-sided Hestenes's method [Hes58]. The algorithm generates an orthogonal matrix V so that AV has orthogonal columns. The columns of AV can then be normalized to 1 so that AV can be written as $U\Sigma$, from which the result follows.

The procedure applies a sequence of plane rotations to A. Let $A_1 = A$ and $V_1 = I$. Choose Q_k to orthogonalize two columns in A_k and let $A_{k+1} = A_k Q_k$ and $V_{k+1} = V_k Q_k$.

Sequentially, the choice of pairs of columns to orthogonalize is made in the order

$$(1,2),(1,3),\ldots,(1,m),(2,3),\ldots(m-1,m).$$

This "sweep" through the matrix must be repeated until all columns are orthogonal. This process turns out to be mathematically identical to the cyclic Jacobi method for the computation of eigenvalues of the matrix XX^T [GvL96]. Wilkinson [Wil62] proves the quadratic convergence properties of this method. In practice, the number of sweeps required is small, typically six to ten [Rut66].

Several improvements to this basic algorithmic structure can be made. Nash [Nas79] pointed out that there is no need to compare the magnitudes of the norms and then choose each plane rotation. Instead, each rotation angle can be chosen so that $\theta \leq \pi/4$.

A further improvement is to maintain a table of the column norms. We call this variant *StdSVD*. Since Hestenes-style algorithms are based on convergence, they are naturally applied as *incremental* algorithms when U, Σ and V matrices are already known. We will call the incremental version of *StdSVD*, *IncSVD*.

Although Hestenes-style algorithms have a higher instruction count than the Golub-Reinsch algorithm (see Table 1), they are more suitable for parallel and incremental computation [Luk80, ZBK94].

3. Incremental SVD

Since SVD computation is expensive, it is useful to design incremental algorithms for (re)computing the SVD of a matrix that has changed, either elementwise or by the addition or deletion of columns and/or rows. Several SVD updating techniques to compute the new SVD when a row (or column) is added to a given matrix are known [CMW$^+$97]. However, little work has been done on updating the SVD of a matrix when its elements have changed, but its size has not. One major application area where this situation arises is when the data represented by the matrix varies over time. If the underlying time-varying data are relatively smooth, the magnitude of the changes in data values over short time periods are likely to be small. Hence the SVD of the matrix at one time instant is likely to be a reasonable approximation to the SVD at the next time instant, and it is possible to exploit this in the computation of the updated SVD.

3.1. Algorithm

We design *StdSVD* by starting from Nash's algorithm [Nas79], and using an array to hold the norm of each column vector. These values are updated after each plane rotation to avoid recomputing the column norms in each iteration of the loop.

When the results of a previous SVD of a slightly different matrix are available, this algorithm can be further improved by starting the first plane rotation with the previous value of V instead of the identity matrix used in Nash's algorithm. We call this algorithm *IncSVD*.

We also add some extra control variables to make the algorithm more flexible. For example, the number of sweeps can be specified as a parameter. Three boolean variables are introduced:

witholdsvd:	true if the previous SVD (V_{old} needed) is available, false otherwise
withu:	true if matrix U is desired, false otherwise
withv:	true if matrix V is desired, false otherwise

maxsweeps denotes the allowable maximum number of sweeps. Parameters *maxsweeps*, τ, ε_M are initialized at the beginning of the program. Given that $n > M$, the matrices U and V are written as:

$$\begin{aligned} U &= (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_m), \\ V &= (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m). \end{aligned}$$

Our algorithm, *IncSVD*, is as follows:

```
1 Initialization previous rotations $V_{old}$ used as starting point if available
    (a) Let $\varepsilon^2 := (3 * 10^{-10})^2$, $\tau^2 := 10^{-20}$, $count := 0$, $sweep := 0$.
    (b) if ( witholdsvd ) $A := AV_{old}$, $V := V_{old}$.
        else if ( withV ) $V := I$.          %previous rotations used
    (c) For $i := 1, 2, \cdots, m$, compute $\|\mathbf{x}_i\|^2$.
    (d) Compute $\delta^2 = \varepsilon^2 \|A\|$.
2 Column orthogonalization
  Repeat until $count = m(m-1)/2$ or $sweep = maxsweep$
    (a) Let $count := 0$, $sweep := sweep + 1$.
    (b) For $(i, j) := (1,2), (1,3), \cdots, (1,m), (2,3), \cdots, (3,4), \cdots, (m-1, m)$ do
        i If $\|\mathbf{x}_j\|^2 < \delta^2$ then let $count := count + 1$.
        ii Else if $\|\mathbf{x}_i\|^2 < \delta^2$, then
            A Exchange $\mathbf{x}_i$ and $\mathbf{x}_j$.
            B If ( withv ) then exchange $\mathbf{v}_i$ and $\mathbf{v}_j$.
        iii Else if $(\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|})^2 < \tau^2$, then
            A If $\|\mathbf{x}_i\|^2 < \|\mathbf{x}_j\|^2$
                (a) Exchange $\mathbf{x}_i$ and $\mathbf{x}_j$.
                (b) If ( withv ) then exchange $\mathbf{v}_i$ and $\mathbf{v}_j$.
            B Else let $count := count + 1$.
        iv Else
            A Compute
                $\alpha = 2\mathbf{x}_i^T \mathbf{x}_j$,
                $\beta = \|\mathbf{x}_i\|^2 - \|\mathbf{x}_j\|^2$, and
                $\gamma = (\alpha^2 + \beta^2)^{1/2}$
```

B Compute $\cos\phi$, $\sin\phi$ based on the sign of β

$$\begin{aligned} \cos\phi &= (\frac{\gamma+\beta}{2\gamma})^{\frac{1}{2}} \ , \ \sin\phi = \frac{\alpha}{2\gamma\cos\phi} \text{ for } \beta > 0 \\ \sin\phi &= (\frac{\gamma-\beta}{2\gamma})^{\frac{1}{2}} \ , \ \cos\phi = \frac{\alpha}{2\gamma\sin\phi} \text{ for } \beta \leq 0 \end{aligned}$$

C Compute the transformed $\mathbf{x}_i$, $\mathbf{x}_j$

$$\begin{aligned} \mathbf{r} &:= \cos\phi\cdot\mathbf{x}_i + \sin\phi\cdot\mathbf{x}_j, \\ \mathbf{x}_j &:= -\sin\phi\cdot\mathbf{x}_i + \cos\phi\cdot\mathbf{x}_j, \ \ \textit{and} \ \ \mathbf{x}_i = \mathbf{r}. \end{aligned}$$

D Update the norm of $\mathbf{x}_i$, $\mathbf{x}_j$

$$\begin{aligned} \|\mathbf{r}\|^2 &:= \cos\phi^2\|\mathbf{x}_i\|^2 + \sin\phi^2\|\mathbf{x}_j\|^2 + \alpha\cos\phi\sin\phi, \\ \|\mathbf{x}_j\|^2 &:= \sin\phi^2\|\mathbf{x}_i\|^2 + \cos\phi^2\|\mathbf{x}_j\|^2 - \alpha\cos\phi\sin\phi, \end{aligned}$$

and

$$\|\mathbf{x}_j\|^2 := \|\mathbf{r}\|^2.$$

E If (*withv*) then

$$\begin{aligned} \mathbf{r} &:= \cos\phi\cdot\mathbf{v}_i + \sin\phi\cdot\mathbf{v}_j, \\ \mathbf{v}_j &:= -\sin\phi\cdot\mathbf{v}_i + \cos\phi\cdot\mathbf{v}_j, \ \ \textit{and} \ \ \mathbf{v}_i = \mathbf{r}. \end{aligned}$$

3 **Computing singular values**

(a) Let $i := 1$.

(b) Repeat until $i > m$ or $\|\mathbf{x}_i\| < \delta$:

  i Let $\sigma_i := \|\mathbf{x}_i\|$ and $i := i + 1$.

  ii If (*withu*) then $\|\mathbf{u}_i\| := \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$

If the previous decomposition is not available, *IncSVD* becomes *StdSVD*.

3.2. Algorithm complexity

Using the number of flops required as approximate estimates of work, we compare the complexities of four methods: the *StdSVD* algorithm, the *IncSVD* algorithm, the Nash method [Nas79] and the Golub-Reinsch method [GvL96]. One sweep of our algorithm takes about $(8n+6m)m^2/2$ flops if the V matrix is desired, and $8nm^2/2$ flops otherwise. Nash's method needs $(10n + 6m)m^2/2$ for one sweep. We assume that Nash's method and our *StdSVD* algorithm take eight sweeps to converge. Our experiments performed on time-varying data show that the 2-sweep *IncSVD* can achieve a highly accurate decomposition (see Table 2); thus

Matrices	*StdSVD*	*IncSVD*	Nash SVD	Golub-Reinsch SVD
U_r, Σ_r, V_r	$32nm^2 + 24m^3$	$8nm^2 + 6m^3$	$40nm^2 + 24m^3$	$14nm^2 + 8m^3$
Σ_r, V_r	$32nm^2 + 24m^3$	$8nm^2 + 6m^3$	$40nm^2 + 24m^3$	$4nm^2 + 8m^3$
Σ_r, U_r	$32nm^2$	$8nm^2$	$40nm^2$	$14nm^2 - 2m^3$
Σ_r	$32nm^2$	$8nm^2$	$40nm^2$	$4nm^2 - 4m^3/3$

Table 1. Computational efficiency of various SVD computation methods

the 2-sweep *IncSVD* algorithm is used for comparison. Table 1 gives the number of flops required by the four methods in four different cases.

In computing the full singular value decomposition, the *StdSVD* algorithm is faster than Nash's method but slower than Golub-Reinsch's method, and the *IncSVD* algorithm is much faster than Nash's method and the Golub-Reinsch method.

3.3. Experimental results

Performance measures. Several measures are defined to evaluate the accuracy of the computed singular values and singular vectors of the *IncSVD* algorithm. In this paper, we use the Frobenius norm for matrices,

$$\|A\| = \|A\|_F = (\sum_{ij} a_{ij}^2)^{\frac{1}{2}}$$

Approximation error. Suppose $\hat{X} \in \Re^{n \times m}$ is an approximation to $X \in \Re^{n \times m}$. For a given matrix norm $\|\cdot\|$ we say that

$$\epsilon_{abs}(\hat{X}) = \|\hat{X} - X\| \tag{22.1}$$

is the *absolute error* in $\hat{X}$ while if $\|X\| \neq 0$ then

$$\epsilon_{rel}(\hat{X}) = \frac{\|\hat{X} - X\|}{\|X\|} \tag{22.2}$$

is the *relative error* in $\hat{X}$.

Non-orthogonality. The non-orthogonality of an n-by-m matrix A is defined as

$$Z(A) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (\mathbf{a}_i^T \mathbf{a}_j)^2, \tag{22.3}$$

where bfa_i is the ith column vector of A.

Rel Change (%)	Abs Change	U RelErr (%)	U AbsErr	V RelErr (%)	V AbsErr	U NonOrtho
12.272	1.3630	0.0000	0.0000	0.0000	0.0000	0.00000
24.266	2.6951	0.0000	0.0000	0.0000	0.0000	0.00000
30.339	3.3695	0.0000	0.0000	0.0000	0.0000	0.00000
27.942	3.1034	0.0000	0.0000	0.0000	0.0000	0.00000
30.137	3.3472	0.0047	0.0001	0.0040	0.0001	0.00000
27.590	3.0643	0.0007	0.0000	0.0004	0.0000	0.00000
14.720	1.6349	0.0000	0.0000	0.0000	0.0000	0.00000
30.488	3.3861	0.0000	0.0000	0.0000	0.0000	0.00000
41.829	4.6457	0.0000	0.0000	0.0000	0.0000	0.00000
24.861	2.7611	0.0000	0.0000	0.0000	0.0000	0.00000
12.654	1.4054	0.0000	0.0000	0.0000	0.0000	0.00000
15.019	1.6680	0.0000	0.0000	0.0000	0.0000	0.00000
10.154	1.1277	0.2999	0.0067	0.2982	0.0067	0.00000
12.732	1.4141	0.0000	0.0000	0.0000	0.0000	0.00000
14.728	1.6357	0.0000	0.0000	0.0000	0.0000	0.00000
10.063	1.1176	0.0000	0.0000	0.0000	0.0000	0.00000
26.869	2.9841	0.0000	0.0000	0.0000	0.0000	0.00000
19.786	2.1975	0.0000	0.0000	0.0000	0.0000	0.00000
31.299	3.4762	0.0000	0.0000	0.0000	0.0000	0.00000
14.359	1.5948	0.0000	0.0000	0.0000	0.0000	0.00000
10.429	1.1583	0.0000	0.0000	0.0000	0.0000	0.00000
13.910	1.5449	0.0000	0.0000	0.0000	0.0000	0.00000
14.992	1.6651	0.0000	0.0000	0.0000	0.0000	0.00000
23.607	2.6219	0.0000	0.0000	0.0000	0.0000	0.00000
16.582	1.8416	0.0000	0.0000	0.0000	0.0000	0.00000
16.920	1.8792	0.0000	0.0000	0.0000	0.0000	0.00000
15.050	1.6715	0.0000	0.0000	0.0000	0.0000	0.00000
10.522	1.1687	0.5777	0.0129	0.5764	0.0129	0.00000
10.869	1.2071	0.0000	0.0000	0.0000	0.0000	0.00000
17.773	1.9739	0.0000	0.0000	0.0000	0.0000	0.00000
21.734	2.4138	0.0000	0.0000	0.0000	0.0000	0.00000
9.746	1.0824	0.0013	0.0000	0.0007	0.0000	0.00000
16.133	1.7918	0.0000	0.0000	0.0000	0.0000	0.00000
19.980	2.2190	0.0000	0.0000	0.0000	0.0000	0.00000

Table 2. Error properties over time steps, *IncSVD* versus *StdSVD*.

Results. For our experimental data, we use performance measures for a set of 31 devices, measured at 254 regular intervals. Each device has 5 measured attributes. The data measured at the tth time, $A(t)$, is therefore a 31×5 matrix where row i represents measured attributes for device i, and column j represents the measurements of the same attribute for different devices.

The singular value decomposition of A is computed directly for the measurements initially. After that, however, singular values and singular vectors are computed incrementally using the 2-sweep *IncSVD*, as well as by *StdSVD* and Golub-Reinsch SVD. In general, the singular values produced by all of these techniques are identical. The singular vectors computed by the *StdSVD* and Golub-Reinsch SVD are also exactly the same. The singular vectors computed by the 2-sweep *IncSVD* are similar, but not identical.

For the experimental data described above, the relative differences in measurements from one time to the next are relatively large (up to 40%), but the maximum approximation error is less than 0.6% for $\hat{U}$ and $\hat{V}$. $\hat{U}$ is orthogonal within the prescribed tolerance (10^{-10}) (see Table 2). This shows that the 2-sweep *IncSVD* is robust enough to get good results.

Although later *IncSVD* decompositions are based on earlier ones, no cumulative error was detected in our experiments. *StdSVD* generally needs 5 or 6 sweeps to generate the fully orthogonal matrix U; hence the 2-sweep *IncSVD* significantly reduces the computational efforts without much loss of accuracy. The main reason why the *IncSVD* works so well is that orthogonality changes between the column vectors are more stable than the numerical changes in the magnitudes of elements in $A(t)$. Therefore the plane rotations in the previous SVD computation are useful as the initial plane rotations in the new SVD decomposition.

4. Parallel SVD

Singular value decomposition is an expensive operation, and it is not surprising that parallel implementations of it have been considered. This has, of course, become of increasing interest recently as SVD is applied to data mining problems whose size is significantly larger than most problems encountered in numerical analysis.

The obvious way to parallelize Hestenes-style algorithms is to execute different iterations of the column normalization loop independently. Any iterations involving disjoint column indexes are independent. For a matrix with m columns, up to m processors can be used, and m (parallel) iterations are required. The key is to find a schedule for the pairs such that as many pairs as possible are considered during each iteration. Many such results have been published [ZB95, GvL96, BL85].

This style of parallelization was developed to fit well with parallel computers connected by particular topologies and with low overhead point-to-point communication primitives. For example, several good schedules that require shifting columns around a ring of processors are known.

Such parallelization techniques do not perform as well on modern parallel computers. The amount of work required for each pair of columns at each step is proportional to n/p, but n/p data must be moved before the next step. Hence the communication overhead has the same magnitude as the computation itself.

5. Parallel cost modelling

In order to discuss the performance of various parallel implementations of SVD, we need a cost model. We will use the cost model associated with Bulk Synchronous Parallelism (BSP) which is both easy to work with and fairly accurate.

BSP programs are sequential compositions of machine-wide operations called *supersteps*. Each superstep consists of three phases: local computation on local data within each processor; global communication; and a barrier sychronization, after which transmitted data becomes visible in the scope of its destination processor. The BSP model applies to both shared-memory and distributed-memory parallel computers, and is available as a library for a full range of platforms. The superstep structure means that BSP is only a good model for programs that have a similar phased structure, but this is fairly common.

The cost of a BSP superstep is the sum of three terms, one for each phase of the superstep. (Conceptually, and even in implementations, these three phases are not overlapped – see [SHM97] for a fuller explanation of why this is the right thing to do.) The first term is the maximum over processors of the local computation time in each one, i.e.

$$\text{computation} = \mathsf{MAX}\, w_i$$

This term is conceptually the simplest, but practically the most difficult to estimate as modern processors drift further and further from simple instruction counting behavior.

The second term is the cost of global communication. Many parallel cost models are based on the cost of individual messages, typically involving a start-up cost and a per-byte transmission cost. The difficulty with such costs is that they don't compose well, because the actual message latency depends on which other messages each message encounters.

The problems of finding the right sum of individual message costs has proved insuperable. The BSP cost model instead observes that the bottleneck in message transmission in modern parallel systems is at the network boundary. Hence, it is the processor that sends and receives the greatest amount of data that will be the bottleneck. Accordingly, the communication cost is modelled by

$$\text{communication} = \mathsf{MAX}\, h_i\, g$$

where h_i is the total traffic into and out of processor i during the global communication phase, and g is an architecture parameter, the *permeability* of the network to traffic. If g is expressed in units of instruction times per transmitted byte, then this cost is in the same units as the previous one.

The cost of the barrier sychronization is given by

$$\text{synchronization} = l$$

also in units of time. The overall cost of a superstep is therefore

$$\text{cost of a superstep} = w + h\, g + l$$

where $w = \mathsf{MAX}\, w_i$ and $h = \mathsf{MAX}\, h_i$.

The cost of an entire program is simply the sum of the cost of its supersteps.

5.1. Broadcasting

The cost of broadcasting an m-by-m matrix directly, an operation we shall use often, is

$$\text{simple broadcast cost} = pm^2\, g$$

However, using a technique that deserves to be more widely known and used, this can be reduced. The technique is called *two-phase broadcast.* In a first step, the matrix is divided into p equal-sized pieces, and each piece is sent to a different processor. In the second step, the processors effect a total exchange of their pieces. The cost of this two-phase technique is:

$$\begin{aligned}\text{two phase broadcast cost} &= (p.m/p\; g + l) + (p.m/p\; g + l) \\ &= 2m^2\, g + 2l\end{aligned}$$

i.e. independent of p. The improved cost of this technique is not a purely theoretical one – the actual improvement on real computers has been verified.

6. Approximate parallel algorithms

We now consider parallel algorithms for SVD that use coarser granularity, and therefore a smaller ratio of communication to computation. Such algorithms are more suitable for modern parallel computers and large matrices.

We begin with a family of approximate algorithms. These are based on the assumption that the matrix contains a large amount of redundancy. This is a plausible assumption in data mining settings since many similar objects are typically captured in such applications.

The justification for these approximate algorithms are Hoeffding (or additive Chernoff) bounds. Suppose that we have a real-valued random variable r with range R, and that we use a sample of size n to compute and estimate, $\bar{r}$, the mean of r. Then with probability $1 - \delta$ the true mean is within ε of the estimated mean where

$$\varepsilon = sqrt\left(\frac{R^2 \ln(1/\delta)}{2n}\right)$$

In other words, as we increase the size of the sample, our uncertainty about the mean decreases.

This bound makes no assumptions about the shape of the distribution involved, so it is a very coarse bound. In practice, values of n much smaller than those it suggests will perform perfectly adequately.

It is also known that the SVD of a matrix is well-behaved with respect to perturbations, that is the difference in the singular values is bounded by the norm of the perturbation of the original matrix [FMM77]. This provides further justification for the usefulness of incremental algorithms, but also provides some reassurance about approximate algorithms to the extent that one partition can be viewed as a perturbation of another. In many data mining settings, this is a reasonable view.

We suppose that the matrix A (of size n-by-m) can be partitioned among p processors so that each processor holds n/p rows. It will be useful to assume that each of these partitions is a uniform random subset of the whole matrix. In practice, this may not be the case, but this is unlikely to cause difficulties unless the matrix has significant internal structure that is not destroyed by the partitioning.

These approximate algorithms are based on computing the SVD for a subset of the rows of the matrix, and using the basis implied by the resulting V matrix as an approximation to the true basis for the transformed space.

Suppose that the partitions of the original matrix, A, are named A_1, A_2, ..., A_p.

Approximate algorithm 1. Algorithm 1 is as follows:

1 Compute the SVD of $A_1 = U_1 \Sigma_1 V_1^T$, only Σ_1 and V_1 are required.

2 Compute $\Sigma = \sqrt{p}\, \Sigma_1$. Compute Σ^{-1} and the product $S = V_1 \Sigma^{-1}$.

3 Broadcast S to the other processors.

4 Compute $U_i = A_i S$

Step 1 creates a transformed m-dimensional space with basis given by V_1. Step 2 approximates the true singular values. If each singular value had been computed independently for each partition then its true value, σ_i, is approximately

$$\sigma_i \approx \sqrt{\Sigma_{j=1}^{p} \sigma_i^2(j)}, \tag{22.4}$$

where $\sigma_i(j)$ is the ith singular value of the jth partition. So the expression in Step 2 is derived by assuming that each Σ_i is identical to Σ_1. Step 4 maps each row of the part of the matrix that is not A_1 into this space using a standard mapping.

The local result is

$$A_i \approx U_i \Sigma V_1^T$$

and the entire decomposition is

$$\begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_p \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ \dots \\ U_p \end{bmatrix} \Sigma V_1^T$$

Notice that this algorithm is identical to latent semantic indexing [DDF90] if the first partition is viewed as a training set and the remainder of the matrix as queries.

The cost of this algorithm is:

Step	Cost
1	$4(n/p)m^2 + 8m^3$
2	$2m + m^2$
3	$2m^2\, g + 2l$
4	$2(n/p)m^2$
Total	$\mathcal{O}((n/p)m^2 + m^3) + \mathcal{O}(m^2)\, g\ +\ \mathcal{O}(1)l$

Hence we have linear speedup in the computation time, at the expense of a communication overhead of $\mathcal{O}(m^2)$. Note, however, that the communication overhead is independent of both n and p and hence the technique scales well.

Approximate algorithm 2. Algorithm 2 increases the flexibility in the way the initial SVD is computed. A subset of the rows of each of the A_i is sent to one of the processors, and this new matrix A_s plays the role of A_1 in the previous version. The advantages to this approach are that it can compensate for some lack of randomness in the distribution of A and that it allows for oversampling, that is using more than n/p rows to compute the initial SVD.

The number of samples selected from each partition can be expressed in the form $s\, n/p^2$. When $s = 1$ this is equivalent to Algorithm 1; when $s < 1$ it undersamples compared to Algorithm 1; and when $s > 1$ it oversamples (which is presumably the interesting case).

The algorithm is:

1 Select $s \times n/p^2$ elements from each processor's partition and send them to processor 1.

2 Call the matrix formed by concatenating these rows, A_s. Compute the SVD of $A_s = U_s \Sigma_s V_s^T$, only Σ_s and V_s required.

3 Execute Algorithm 1 using Σ_s in place of Σ_1 and V_s in place of V_1.

The cost of this algorithm is:

Step	Cost
1	$s(n/p)\; m\; g + 2l$
2	$4s\; (n/p)\; m^2 + 8m^3$
3	as before
Total	$O(s(n/p)m^2 + m^3) + O(s(n/p)m + m^2)\; g \;+\; O(1)l$

The computation cost for Algorithm 2 is similar to that of Algorithm 1 except that it can be altered by choosing a value for s different from 1. The communication cost, however, increases because of the extra step required to gather the rows for A_s. As we shall see below, this extra cost is not too important since good results can be achieved when n/p is only a small multiple of m.

6.1. Performance results

We now show the difference in the computed SVDs between the exact solution and those produced by the approximate algorithms.

Two kinds of random matrices are used in our experiments. The first are matrices whose elements are (uniformly) random values between 0 and 1. The second are matrices with specified singular values generated according to a geometric distribution:

$$\sigma_i = \sigma^{\frac{i-1}{p-1}}, \quad i = 1, 2, \ldots, \quad p = \min(n, m) \tag{22.5}$$

such that $\|A\| = \sigma_1 = 1$ and the singular values decay gradually. This distribution of singular values are widely deployed in comparative experiments [Jia, Han88]. Relative errors $\epsilon_{rel}(\hat{U})$, $\epsilon_{rel}(\hat{\Sigma})$ and $\epsilon_{rel}(\hat{V})$ are computed.

We use matrices of size 4096 × 64 with condition number 100. The computational precision is 10^{-10}.

Results for Approximate algorithm 1. Experiment 1 uses a random matrix while Experiment 2 uses a random matrix with prescribed singular values. The distribution of singular values for these two matrices are plotted in Figure 1 and 2. Experimental results are shown in Tables 3 and 4.

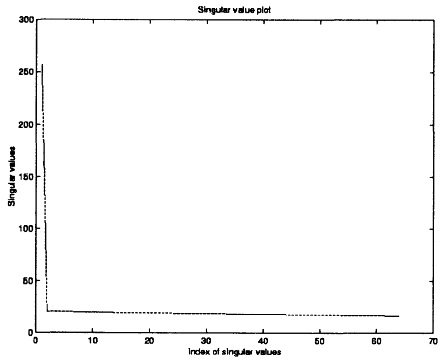


Figure 1. Singular values for a 4096 × 64 random matrix

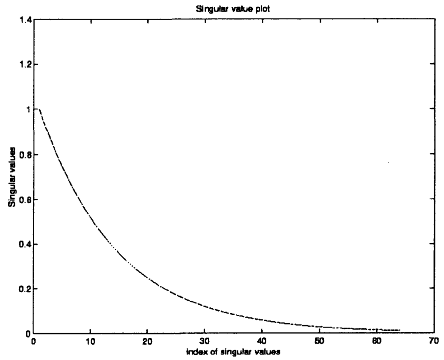


Figure 2. Singular values for a 4096 × 64 random matrix with geometrically distributed singular values

Partition size	# Processors	$\epsilon_{rel}(\hat{U})$	$\epsilon_{rel}(\hat{\Sigma})$	$\epsilon_{rel}(\hat{V})$
64	64	16.1258	0.2469	1.4116
128	32	1.7233	0.1552	1.3927
256	16	1.5082	0.0952	1.4028
512	8	1.4204	0.0580	1.3757
1024	4	1.4028	0.0304	1.3848
2048	2	1.3972	0.0127	1.3920

Table 3. Approximation error for a 4096 × 64 random matrix using Approximate algorithm 1

From the tables, it can be seen that the relative errors are fairly large when the partition size (n/p) is about the same magnitude as m, but decrease as partition sizes become only slightly larger, remaining

Partition size	# Processors	$\epsilon_{rel}(\hat{U})$	$\epsilon_{rel}(\hat{\Sigma})$	$\epsilon_{rel}(\hat{V})$
64	64	40.2189	0.0658	1.3431
128	32	1.6095	0.0423	1.3106
256	16	1.3371	0.0336	1.2222
512	8	1.2068	0.0291	1.1617
1024	4	0.9860	0.0149	0.9654
2048	2	0.8133	0.0085	0.8075

Table 4. Approximation error for a 4096 × 64 random matrix with specified singular values using Approximate algorithm 1

flat after that. The very small approximation error for singular values verifies our assumption that the variance among the data are evenly distributed in each partition (see Equation 22.4).

The average percentage error *per element* is in the range $10^{-3} - 10^{-2}$ for U and V matrices, which are usually the results of interest in data mining applications.

Results for Approximate algorithm 2. We randomly sample each partition to construct the new matrix A_s. Experiment 3 uses the same data as Experiment 1 and Experiment 4 uses the same data as Experiment 2. Experiment results are given in Tables 5 and 6.

Partition size	# Processors	$\epsilon_{rel}(\hat{U})$	$\epsilon_{rel}(\hat{\Sigma})$	$\epsilon_{rel}(\hat{V})$
64	64	111.4076	0.2469	1.4099
128	32	1.7108	0.1488	1.4102
256	16	1.5163	0.0963	1.4091
512	8	1.4129	0.0553	1.3752
1024	4	1.4027	0.0311	1.3819
2048	2	1.4280	0.0118	1.4224

Table 5. Approximation error for a 4096 × 64 random matrix using Approximate algorithm 2

The results are similar to those of Algorithm 1 which is not surprising given that the original matrices are generated randomly.

6.2. Choosing only some singular values

In data mining applications, it is common to focus on the first k singular vectors as a dimensionality reduction technique. This has performance implications, since matrices that are m-by-m in the basic al-

Partition size	# Processors	$\epsilon_{rel}(\hat{U})$	$\epsilon_{rel}(\hat{\Sigma})$	$\epsilon_{rel}(\hat{V})$
64	64	13.9569	0.0929	1.4139
128	32	1.6948	0.0592	1.3593
256	16	1.2450	0.0276	1.1209
512	8	0.9138	0.0308	0.8627
1024	4	0.8858	0.0145	0.8713
2048	2	0.7727	0.0086	0.7624

Table 6. Approximation error for a 4096 × 64 random matrix with specified singular values using Approximate algorithm 2

gorithms become m-by-k. Hence the $\mathcal{O}(m^2)$ terms in the complexities become $\mathcal{O}(mk)$ terms.

However, it might be expected that ignoring some columns of the V matrices might increase the error in approximations. We did some experiments to find the relationship between the average error per element and the size of k. Figures 3 — 14 give the results for the four experiments.

From these figures we can see that the errors are smaller, across the board, for matrices with defined singular values. These are much more typical of matrices seen in data mining applications than random matrices are, so these errors are what might be expected in practice.

The relative errors in both U and V generally decrease as k increases. However, for random matrices there are two anomalies: the spike for small values of k and the increasing error as k approaches m. We hypothesize that the spike for small k is a consequence of the fact that almost all of the variation in the data is captured by the first dimension and first singular value. Adding a few more dimensions simply adds noise to this approximation. Adding further dimensions and singular values improves the approximation, but often it is still not as accurate as the 1-dimensional approximation. The error rates for the singular values depend heavily on the size of the partition or sample used for random matrices. The increasing error as k approaches m may be caused by different column orderings in V in different partitions because the singular values are so close (i.e roundoff errors start to have some effect).

Overall these results suggest that approximations for the U matrix with error rates per element below 5×10^{-5} can be achieved using only a small fraction of the singular values, that is by choosing a fairly small k.

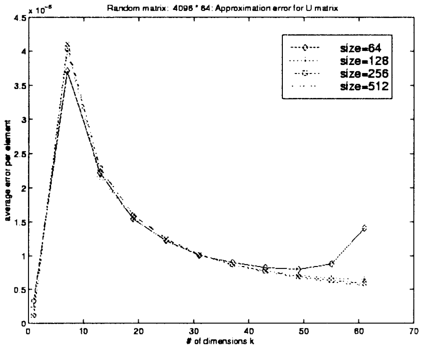


Figure 3. Average error per element vs. k for U in Experiment 1

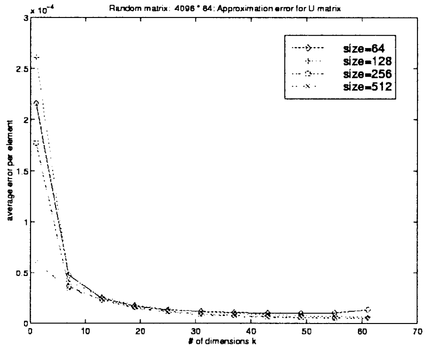


Figure 4. Average error per element vs. k for U in Experiment 3

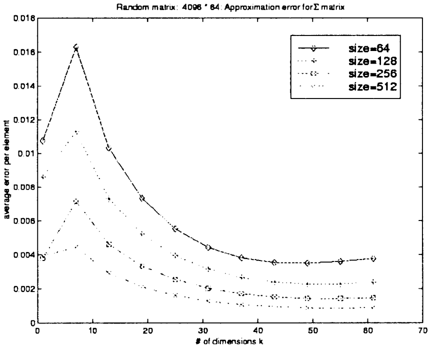


Figure 5. Average error per element vs. k for Σ in Experiment 1

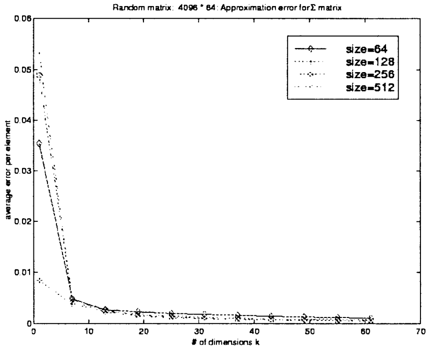


Figure 6. Average error per element vs. k for Σ in Experiment 3

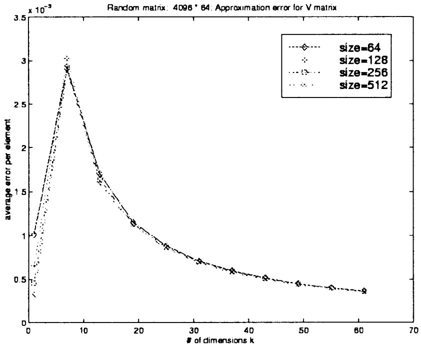


Figure 7. Average error per element vs. k for V in Experiment 1

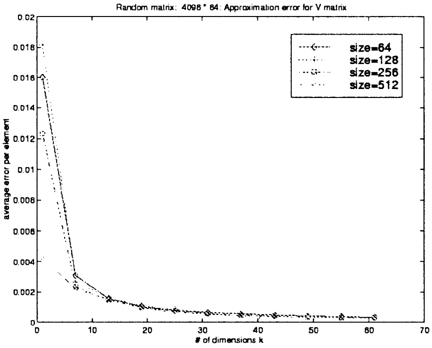


Figure 8. Average error per element vs. k for V in Experiment 3

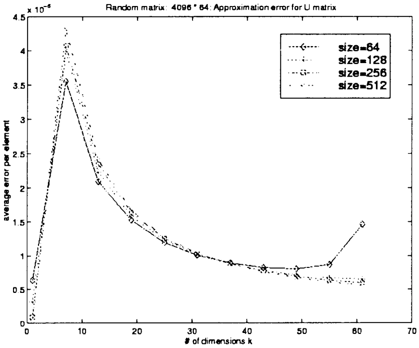


Figure 9. Average error per element vs. k for U in Experiment 2

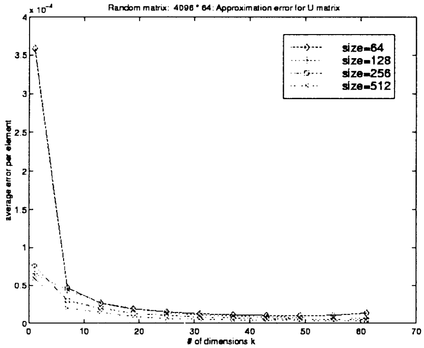


Figure 10. Average error per element vs. k for U in Experiment 4

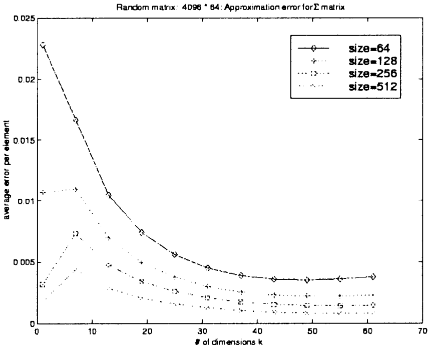


Figure 11. Average error per element vs. k for Σ in Experiment 2

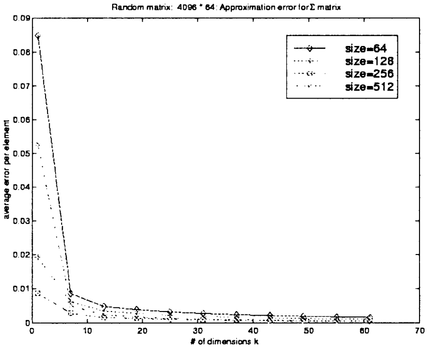


Figure 12. Average error per element vs. k for Σ in Experiment 4

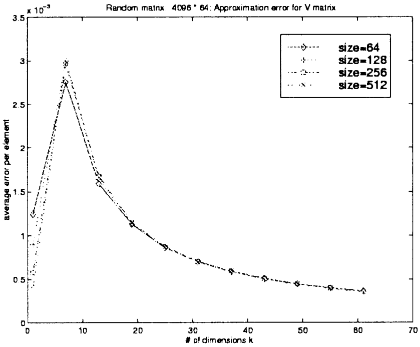


Figure 13. Average error per element vs. k for V in Experiment 2

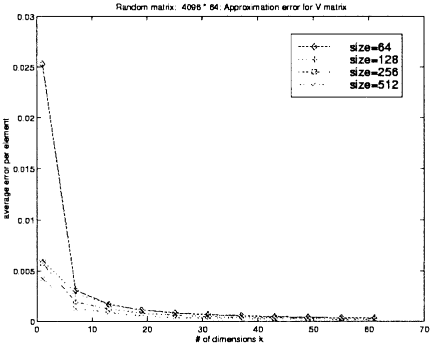


Figure 14. Average error per element vs. k for V in Experiment 4

7. A parallel and distributed algorithm

We now consider a true parallel SVD. In some data-mining settings, the distribution of the data cannot be controlled. For example, some parts of the dataset may be distributed geographically because of where they are captured. It may not be possible to gather the entire dataset to one place to carry out a sequential SVD because the whole dataset is simply too large to store in one place. It may equally well not be possible to use the approximate parallel algorithm because the subset of the data at each site is not a uniform subset of the entire dataset. Hence an algorithm that can compute the global SVD no matter what the partitioning of the input dataset is required.

7.1. Basis

There are important relationships between the singular value decomposition of a matrix A and the symmetric matrices $A^T A$ and AA^T. Indeed, if

$$U^T AV = diag(\sigma_1, \dots, \sigma_n)$$

is the SVD of $A \in \Re^{n \times m} (n \geq m)$ then

$$V^T(A^T A)V = diag(\sigma_1^2, \dots, \sigma_m^2) \in \Re^{m \times m} \tag{22.6}$$

and

$$U^T(AA^T)U = diag(\sigma_1^2, \dots, \sigma_m^2, \underbrace{0, \dots, 0}_{n-m}) \in \Re^{n \times n}. \tag{22.7}$$

$A^T A$ is both symmetric and positive definite. These equations suggest us to adapt the singular value problem to the symmetric eigenproblem. That is

1 Form $C = A^T A$.

2 Use any SVD algorithm for symmetric eigenproblems to compute $V_1^T C V_1 = diag(\sigma_i^2)$.

3 Obtain $\Sigma = diag(\sigma_i)$.

4 Obtain U by computing $AV\Sigma^{-1}$.

The formation of $A^T A$ can lead to a loss of information. The situation is not quite so bad here, since the original A is used to compute U.

7.2. Parallel algorithm

Suppose processor i holds partition A_i. A^TA is equal to

$$A^TA = \begin{bmatrix} A_1^T & A_2^T & \dots & A_p^T \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_p \end{bmatrix} = \sum_{i=1}^{p} A_i^T A_i$$

within roundoff error. Using the method in Section 7.1 to decompose A^TA, we can get U from the computed V and Σ :

$$U = \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_p \end{bmatrix} V\,\Sigma^{-1}.$$

Thus, an implementation of parallel SVD is as follows:

1 Compute ${A_i}^T A_i$ at processor i in parallel.

2 Send ${A_i}^T A_i$ to a given processor s and sum them together to $A^TA = C$.

3 Use a Golub-Reinsch SVD to compute $V^TCV = diag(\sigma_i^2)$ at processor s.

4 Obtain $\Sigma = diag(\sigma_i)$.

5 Broadcast V and Σ to each processor.

6 Obtain U_i by computing $A_i V \Sigma^{-1}$.

The cost of this algorithm is:

Step	Cost
1	$2(n/p)m^2$
2	$m^2p\,g + pm^2 + l$
3	$12m^3$
4	m
5	$4m^2g + 2l$
6	$2(n/p)m^2$
Total	$\mathcal{O}((n/p)m^2 + m^3) + \mathcal{O}(pm^2)\,g + \mathcal{O}(1)l$

Such an algorithm is an alternative to techniques such as principal component analysis [KHKJ00] and wavelet bases [HK99] for distributed data mining problems.

7.3. Performance results

This parallel algorithm is accurate in a strict sense since the decomposition matrices are equal to the those from sequential computation within the limits of roundoff error. Experiments on matrices with random entries, and with chosen singular values, of different sizes (from 4000 to 1000), and using different number of processors (from 2 to 64) produced identical results (within the precision bound of 10^{-10}) to sequential SVD.

8. Summary

The performance of the three parallel algorithms described in the paper are:

Algorithm	Computation	Communication
Approximate 1	$O((n/p)m^2 + m^3)$	$O(m^2)$
Approximate 2	$O(s(n/p)m^2 + m^3)$	$O(s(n/p)m + m^2)$
Parallel	$O((n/p)m^2 + m^3)$	$O(pm^2)$

In each case, the computation cost is divided evenly across the processors, leading to p-fold speedup. The communication cost is lowest for Approximate algorithm 1 because it broadcasts only a single m-by-m matrix. Approximate algorithm 2 has a slightly larger communication cost because of the sample gathering step, although the algorithm is sufficiently accurate in practice when n/p is only slightly larger than m. The exact parallel algorithm has the largest communication cost and so should only be used when the initial distribution of the data prevents the use of one of the other algorithms. As we have indicated, this is likely to happen in distributed data mining settings.

References

[BC97] Lisa Ballesteros and W. Bruce Croft. Phrasal translation and query expansion techniques for cross-language information retrieval. In *Research and Development in Information Retrieval*, pages 84–91, 1997.

[BF96] Michael W. Berry and Ricardo D. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numerical linear algebra with applications*, 3(4):301–327, 1996.

[BL85] R. P. Brent and F. T. Luk. "The solution of singular value amd symmetric eigenvalue problems on multiprocessor arrays". *SIAM Journal of Science and Statical Computation*, 6:69–84, 1985.

[CMW$^+$97] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. "An eigenspace update algorithm for image analysis". *Graphical Models and Image Processing*, 59(5):321–332, 1997.

[DDF90] S. Deerwester, S. T. Dumais, and G. W. Furnas. "Indexing by latent semantic analysis". *Journal of the Society for Information Science*, 41:391–407, 1990.

[FMM77] G.E. Forsythe, M.A. Malcolm, and C.B. Moler. *Computer Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

[GvL96] G.H. Golub and C.F. van Loan. *Matrix Computations.* Johns Hopkins University Press, 3rd edition, 1996.

[Han88] Per Christan Hansen. Reducing the number of sweeps in hestense' method. In E. F. Deprettere, editor, *SVD and Signal Processing: Algorithms, Applications and Architecture*, pages 357–368. North-Holland, 1988.

[Hes58] M. R. Hestenes. "Inversion of matrices by biorthogonalization and related issues". *SIAM Journal*, 6:51–90, 1958.

[HK99] D Hershberger and H. Kargupta. Distributed multivariate regression using wavelet-based collective data mining. *Parallel and Distributed Computing, Special Issue on Data Mining*, 1999.

[Jia] Fan Jiang. Efficient singular value decomposition via improved document sampling.

[KHKJ00] H. Kargupta, W. Huang, Krishnamurthy, and E. Johnson. Distributed clustering using collective principal component analysis. In *ACM SIGKDD-2000 Workshop on Distributed and Parallel Knowledge Discovery*, 2000.

[Luk80] F. T. Luk. "Computing the singular value decomposition on the ILLIAC IV". *ACM Transactions on Mathematical Software*, 6:524–539, 1980.

[Nas79] J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation.* John Wiley & Sons, 1979.

[Pry98] Michael H. Pryor. The effects of singular value decomposition on collaborative filtering. Technical Report PCS-TR98-338, Yale University, 1998.

[Rut66] H. Rutishauser. "The Jacobi method for real symmetric triangular matrices". *Numerical Mathematics*, 9:1–10, 1966.

[SHM97] D.B. Skillicorn, J.M.D. Hill, and W.F. McColl. Questions and answers about BSP. *Scientific Programming*, 6(3):249–274, 1997.

[Wil62] J. H. Wilkinson. "Note on the quadratic convergence of the cyclic Jacobi process". *Numerical Mathematics*, 4:296–300, 1962.

[ZB95] B. Zhou and R. Brent. The parallel implementation of the one-sided Jacobi algorithm for singular value decompositions. In *Proc. of 3rd Euromicro Workshop on Parallel and Distributed Processing*, pages 401–408, January 1995.

[ZBK94] B. B. Zhou, R. P. Brent, and M. Kahn. A one-sided jacobi algorithm for the symmetric eigenvalue problem. In *Proc. of 3rd Parallel Computing Workshop*, November 1994.

[ZKea98] W. Zhao, A. Krishnaswamy, and R. Chellappa et al. "Discriminant analysis of principal components for face recognition". In H. Wechsler, V. Bruce, and T. Huang et al., editors, *Face Recognition: From Theory to Applications*, pages 73–85. Springer-Verlag, 1998.

Chapter 23

MINING HIGH-DIMENSIONAL SCIENTIFIC DATA SETS USING SINGULAR VALUE DECOMPOSITION

Ekaterina Maltseva, Clara Pizzuti, and Domenico Talia

Abstract Clustering is an undirected knowledge discovery technique based on the partitioning of large sets of data objects into homogenous groups. All objects contained in the same group have similar characteristics. Grouping multivariate data is a difficult data mining task when no domain knowledge on data structure is available. In this chapter we describe the use of a well known linear projection technique, called Singular Value Decomposition (SVD) to discover clusters. SVD is an optimal dimensionality reduction method that projects a multi-dimensional pattern space into a subspace that preserves the character of data. The plot of the 2- or 3-dimensional points gives the human user a guide to discover the presence of homogeneous groups (clusters) in the data set. A user, by inspecting the graphical 2- or 3-dimensional representations, may identify groups on the basis of space density and define threshold values to separate clusters. Experimental results on real scientific data sets assess the quality of the clustering obtained.

Keywords: Singular Value Decomposition, clustering, visual data mining, high dimensional data sets

1. Introduction

Clustering is a data mining task [FS96] for unsupervised classification that consists in partitioning large sets of data objects into homogeneous groups [Eve77, DH73, KR90, JD88]. All objects contained in the same group have similar characteristics. Clustering is an undirected knowledge discovery technique that is used to discover meaningful structures in data when no predefined knowledge is available. Generally, an object is described by a set of m attributes (features) and it can be represented by an m-dimensional vector, called *pattern*. Thus, if a data set contains

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 425–438.

© 2001 *Kluwer Academic Publishers.*

n objects, it can be viewed as an $n \times m$ *pattern matrix*. Rows of the matrix correspond to patterns and columns to features. Each pattern can then be considered as a point in an m-dimensional space, where each of the m attributes is one of the axis of the space.

Clusters can be described as regions of this space containing points which are close to each other. This view of clusters directly reflects the process we use to detect clusters visually in one, two or three dimensions. However, clustering algorithms must be able to detect such groups in multidimensional spaces. An approach to reduce the intrinsic complexity of finding groups in multivariate data is *linear projection* [JD88]. It consists in mapping a set of m-dimensional patterns onto a d-dimensional space, where $d < m$.

A linear projection expresses the d features as linear combination of the original m features [JD88]:

$$y_i = \mathcal{H}x_i \; i = 1, \ldots, n$$

where y_i is a d vector, $\mathcal{H}$ is a $d \times m$ matrix and x_i is an m vector. A main characteristic of linear projection algorithms is that they retain the structure of data. If an accurate two or three-dimensional approximation of the pattern matrix can be obtained, a visual perception of the existing groups could be used for many purposes, such as validating the results of clustering algorithms, providing initial seed information for partitional clustering algorithms, or trying to define threshold values for the projected features in order to assign the original patterns to the discovered groups on the basis of the values of the projected features.
The main goal of this work is to use a well known linear projection algorithm named *Singular Value Decomposition* (*SVD*), to discover groups in multivariate data and to evaluate the quality of clustering results on real scientific data sets from the UCI Machine Learning repository [BM] for which membership class is known. The experiments we have performed on nine different data sets demonstrated that the SVD technique has been useful in obtaining a good clustering result for at least five data sets and showed how the SVD approach could be useful for data clustering. Because of space limits we describe here only the results obtained on two data sets: *Wisconsin breast cancer*, and *dermatology*. The chapter is organized as follows. Section 2 introduces the SVD technique and section 3 discusses in detail the experiments we performed on the two data sets. Section 4 summarizes the results and concludes the chapter.

2. Singular Value Decomposition

SVD is a powerful technique in matrix computation and analysis that has been introduced by Beltrami in 1873 [Bel73]. More recently it has been used in several applications such as solving systems of linear equations, linear regression [Str80], pattern recognition [DH73], statistical analysis [Jol86], data compression [Kor97] and matrix approximation [Pre92].

Singular Value Decomposition is based on linear algebra concepts; thus, some preliminary definitions are given.

Let $\mathcal{R}$ denote the set of real numbers and X a matrix $X \in \mathcal{R}^{n \times m}$. The *transpose* of a matrix X, denoted X^T, is the matrix obtained from X by exchanging the rows with the columns.

A matrix X is said *orthogonal* if its columns are mutually orthogonal vectors of length 1, that is $x^T x = 1$, where x denotes a column of X. This means that $X^T X = I$, the identity matrix.

A set of vectors $\{x_1, \ldots, x_n\}$ is said *linearly independent* if

$$c_1 x_1 + \ldots c_n x_n = 0$$

implies that $c_i = 0$ for $i = 1, \ldots, n$. Otherwise $\{x_1, \ldots, x_n\}$ is said *linearly dependent.*

The *rank* r of a matrix X is the highest number of linearly independent rows (or columns).

For a square matrix X, a unit vector u and a scalar σ that satisfy $Xu = \sigma u$ are called *eigenvector* and *eigenvalue* of X respectively.

A *singular value decomposition* of an $n \times m$ matrix X is any factorization of the form

$$X = U \times \Lambda \times V^T$$

where U is an $n \times n$ orthogonal matrix, V is an $m \times m$ orthogonal matrix and Λ is an $n \times m$ diagonal matrix with $\lambda_{ij} = 0$ if $i \neq j$. It has been shown that there exist matrices U and V such that the diagonal elements of Λ are sorted: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$. The diagonal elements λ_i are called *singular values* of X and it has been shown that they are the square root of the eigenvalues of the matrix $X^T X$.

The decomposition can equivalently be written as

$$X = \lambda_1 u_1 \times v_1^t + \lambda_2 u_2 \times v_2^t + \ldots + \lambda_m u_m \times v_m^t$$

where u_i and v_i are column vectors of the matrices U and V respectively, λ_i are the diagonal elements of Λ, and it is known as *spectral decomposition* [Jol86]. *SVD* reveals an important information about the rank of the matrix X. In fact, if $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq \lambda_{r+1} = \ldots = \lambda_m = 0$ then r is the rank of X [GVL89].

Geometrically this factorization defines a rotation of the axis of the vector space defined by X where V gives the directions, Λ the strengths of the dimensions and $U \times \Lambda$ the position of the points along the new axis.

Intuitively, the U matrix can be viewed as a similarity matrix among the rows of X, i.e. the objects of the data set, the V matrix as a similarity matrix among the columns of X, i.e. the features that describe an object, the Λ matrix gives a measure of how much the data distribution is kept in the new space [JD88].

In the data mining area, *SVD* can be used to identify clusters by analyzing the U and V matrices. By visualizing the V matrix in no more than 3 dimensions it can give a guidance on the number of potential clusters. In fact, as already observed, each row of V corresponds to an attribute of the patterns. If groups of attributes are similar and, thus, are plotted in the same region, they will probably contribute to form clusters characterized by these features. By visualizing the matrix $U \times \Lambda$ and considering only the first d dimensions, where $d \leq 3$, we obtain a compressed representation of the X matrix that approximates it optimally. The d kept terms are known as the *principal components* [Jol86]. If

$$\sum_{i=1}^{d} \lambda_i / \sum_{i=1}^{m} \lambda_i \geq 0.95$$

then the original pattern matrix X is well approximated by the first d terms of the spectral decomposition [JD88].

In the following section the experiments carried out by using the *SVD* technique for the analysis of different data sets are reported and discussed. Furthermore, tentative assignments of points to the discovered clusters is done by defining threshold values for the attributes. To assess the validity of the method, the confusion matrix for the two data sets is also given.

3. Experiments

In this section we describe the results obtained by using the SVD technique for the analysis of two scientific data sets from the UCI Machine Learning repository [BM]: *Wisconsin breast cancer*, and *dermatology*.

Wisconsin Breast Cancer. This data set contains 699 instances of patients that compose two groups of people: 458 benign and 241 malignant. Each instance is characterized by 9 attributes that express the features of the cells (like dimension, form, nucleolus number, etc.) and whose values range over the interval [1..10]. The use of SVD on this data set has shown good clustering results.

By visualizing the first two columns of the matrix V (figure 1), 5 attributes are located close to each other, while the other 4 are placed far from each other. Considering only the first dimension of the $U \times \Lambda$

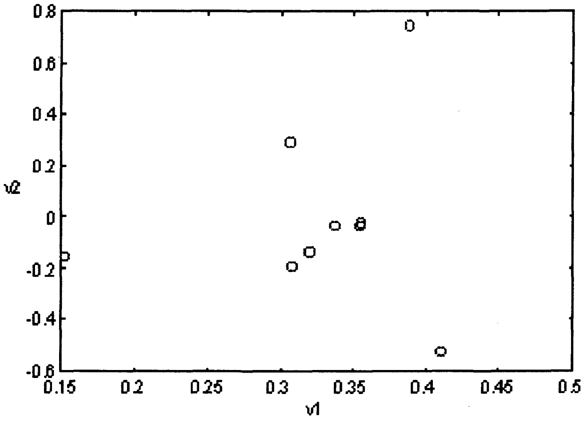


Figure 1. 2-D visualization of the V matrix.

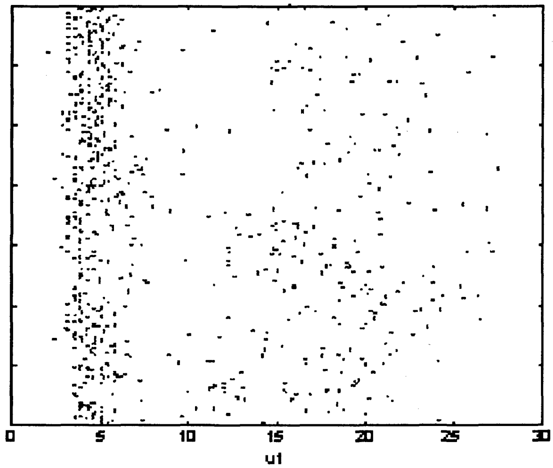
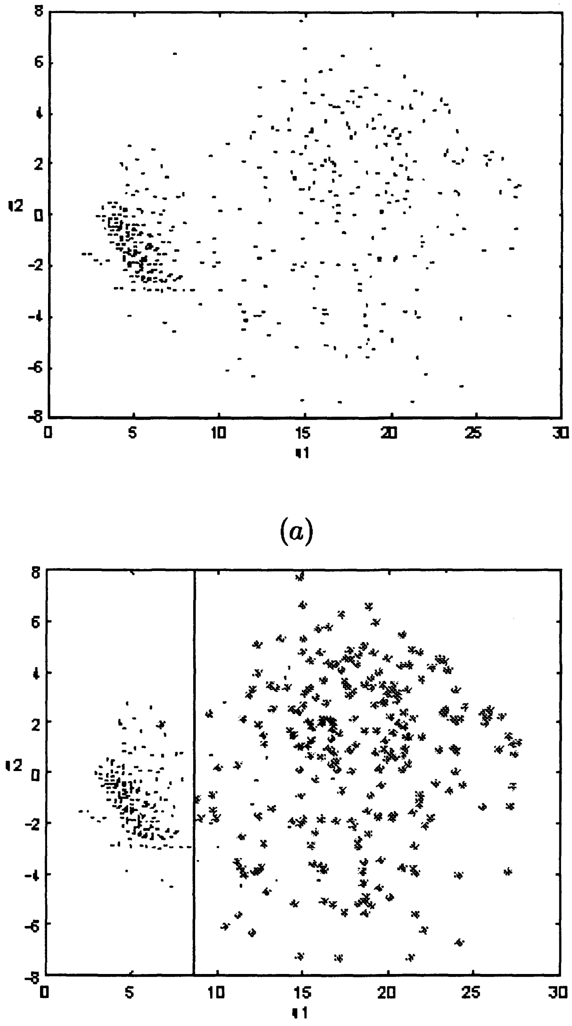


Figure 2. Visualization of the first column of the $U \times \Lambda$ matrix.

matrix (figure 2) we verified the presence of two groups that are separable in $u_1 = 9$.

This hypothesis is confirmed by the two-dimensional visualization of $U \times \Lambda$ (figure 3) where two groups of points can be observed: on the left there is a high concentration of points while of the right part numerous sparse points appear.

As mentioned before, the two-dimensional visualization of the $U \times \Lambda$ matrix (figure 3) for this subset of data shows that it is possible to



(a)

(b)

Figure 3. 2-D visualization of the first two columns of the $U \times \Lambda$ matrix: unique symbol (a) and one symbol per class (b). Benign: ".", malignant: "*".

Table 1. Confusion matrix for the Wisconsin breast cancer data set.

Benign	**Malignant**	← *classified as*
440	18	Benign
1	240	Malignant

identify two groups in it. For a better reading of this result, in figure 1.3(b) the items belonging to different clusters are visualized with two different symbols. The definition of the threshold $u_1 = 9$ allows for the assessment of our approach by estimating the error of SVD classification with respect to real classes. By using this threshold, we obtained the confusion matrix shown in table 1 that shows the difference between the actual classes and the clusters suggested by the SVD.

From table 1 we can see that 19 of the 699 instances are misclassified, thus SVD helped us to obtain a very good clustering of this data set with an accuracy equals to 0.97 and a very low error of 0.3. Furthermore, we verified that the best useful visualization for this data set is two-dimensional.

Dermatology. This data set contains 358 instances each one consisting of 33 attributes. Each instance contains the clinical and histopathological information about diagnosis of patients that suffer of eryhemato-squamous diseases. All the patients should be grouped into six clusters.

By visualizing the first two columns of V (figure 4 (b)) we observed five or six possible groups of values so that the attributes seem to be separable into about five or six groups. This is useful information for the clustering process.

By considering only one dimension of $U \times \Lambda$ we verified that by visualizing the first or the third column, the projection of items along these dimensions gives a uniform distribution, so we cannot distinguish any grouping. However, by projecting the data set along the second column of $U \times \Lambda$ (figure 4 (a)) it is possible to distinguish at least three clusters. Notice that the points where $-1 \le u_2 \le 2$ constitute only one group along the u_2 axis.

Then, by using a two-dimensional visualization of $U \times \Lambda$ (figure 5 (a)) we can see three clusters of items. Indeed the central cluster contains the items of four actual classes (figure 5 (b)). However, at this stage we were not able to know if each or some of these clusters contained more than a

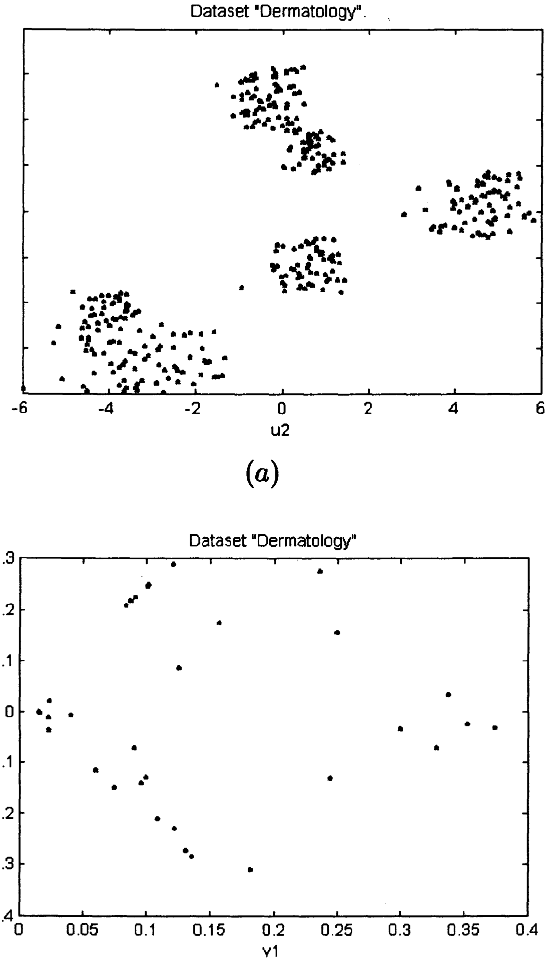
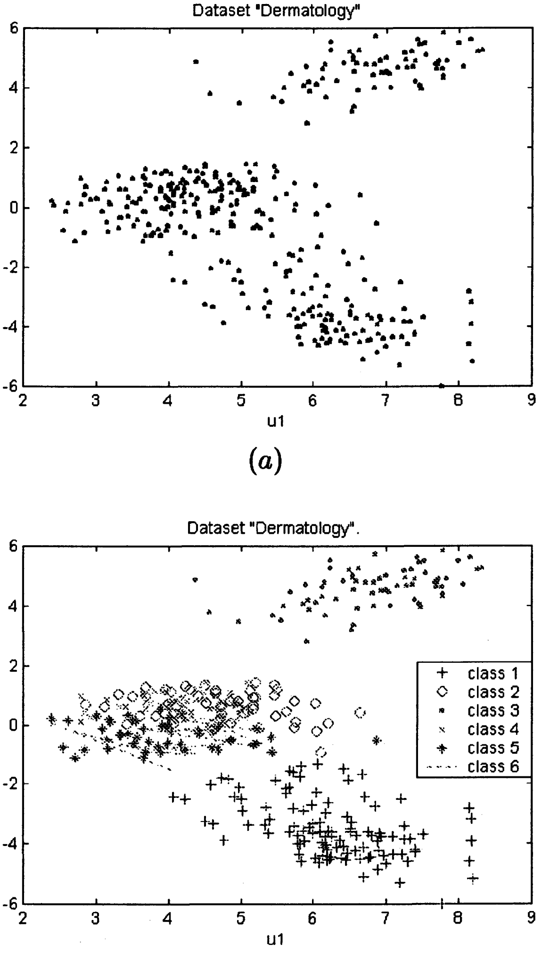


Figure 4. (a) 1-D visualization of $U \times \Lambda$, (b) V matrix of the dermatology data set.



(*a*)

(*b*)

Figure 5. 2-D visualization of the dermatology data set: (a) unique symbol, (b) one symbol per class.

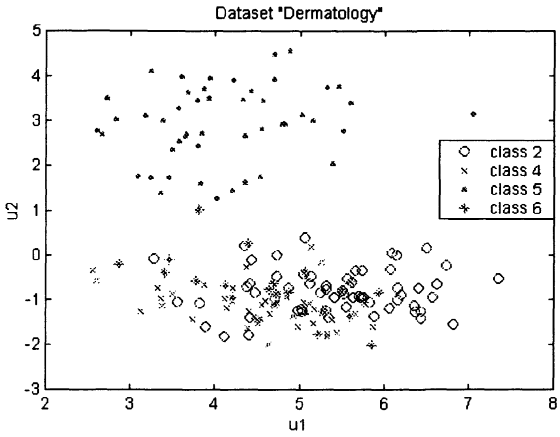


Figure 6. 2-D visualization of the subset of items of classes 2, 4, 5, and 6.

class of items. To try to discover whether the clusters already identified contain different classes of items, we used a step-by-step approach based on the use of SVD on each one of the three clusters.

The SVD analysis on the 111 items that compose the top-right cluster shown in figure 5 (a) resulted in a uniform distribution of the projection of items along the first two columns of the $U \times \Lambda$ matrix. The same result has been obtained applying SVD to the 71 items of the bottom-right cluster of figure 5 (a). This means that SVD suggests that the two clusters are homogeneous so they do not contain sub-clusters, as really occurs. Then we applied SVD to the 176 items that compose the central cluster. In the figure 6 we can notice two clusters well separated by the threshold value $u2 = 1$. Although, in the original figure the items were represented by the same symbol, in figure 6 (and in the next two) we used different symbols for items of different classes to allow the reader to do a better interpretation of the SVD results.

Actually, these results confirmed our hypothesis about the usefulness of SVD in discovering other clusters in subsets of data after the deletion of items that belong to the clusters that in the previous steps have been clearly identified. Proceeding step by step, we applied SVD to the 128 items that compose the bottom cluster of figure 6. For them figure 7 shows a 2-D visualization based on the first two columns of $U \times \Lambda$.

Two groups of items are showed in it separated by the threshold value $u2 = -1.5$. The bottom group, in particular, contains the items of class 6, whereas the top group consists of items of different classes. In the final step, we applied SVD on the 108 items of this group to verify if SVD

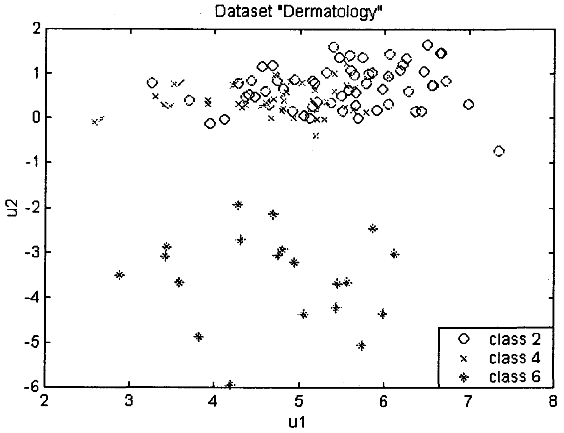


Figure 7. 2-D visualization of the subset of items of classes 2, 4, and 6.

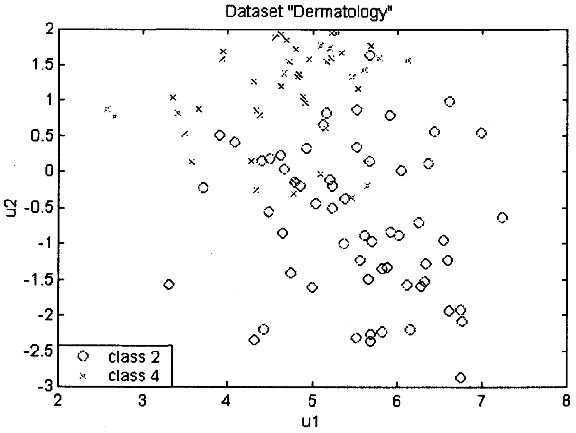


Figure 8. 2-D visualization of the subset of items of classes 2 and 4.

is able to discover the two clusters that it contains indeed. As we can see in figure 8, that shows the plot of items using the first two columns of $U \times \Lambda$, SVD was not able to give an evident separation between the items of classes 2 and 4. We tried also to use different components of $U \times \Lambda$ and different dimensions for representing this data sets obtaining the same result.

In summary, at the beginning SVD discovered three clusters on this data set. Then by a step-by-step deeper analysis of each of these clusters, SVD supported us in discovering 5 of the 6 real classes in which the items of this data set are grouped. This result could be effectively used as input information for an eventual supervised classification step performed on the data set with other data mining tools. This experiment is very positive because it demonstrates that SVD also offers a user helpful clustering support in the analysis of complex data sets whose clustering cannot be discovered in a single step.

4. Conclusions

In this chapter we reported the experiences using the SVD technique for the analysis of different high-dimensional real scientific data sets. In particular, we discussed the use of SVD on the *Wisconsin breast cancer* and *dermatology* databases. However, we tested this methodology also on the following databases of the UCI Machine Learning repository: *ionosphere*, *ecoli*, *iris plant*, *Wisconsin diagnostic breast cancer*, *liver disorders*, and *congressional voting records*. The often positive and sometime negative results that we obtained in using SVD in such data domains showed that this technique could be useful in unsupervised clustering of data. In particular, it could be coupled with visual data mining and classification techniques to offer a complete knowledge discovery methodology for multi-dimensional scientific data sets.

The output of the SVD technique is in the matrices Λ, V, and U. Thus, when SVD is used for clustering, a detailed analysis of these matrices is necessary to discover in them values that suggest how data can be visualized and interpreted to extract the structure of clusters from data. The values of the diagonal matrix Λ indicates the number of attributes that could be useful to consider for getting the best approximation of data. The higher the value of an element of Λ, the more significant the corresponding column of U. On the basis of our experiments, the 1-D visualization of $U \times \Lambda$ allows for selecting the column that permits a better identification of distinct clusters. However, a single dimension is not ever sufficient; in general we experienced that 2-D representations of $U \times \Lambda$ give the best way to identify clusters. Furthermore, a step-by-step approach based on first applying the SVD on subsets of data that compose the clusters discovered at the previous step, and then analyzing the $U \times \Lambda$ matrix for these subsets, allowed us to discover clusters that previously were not visible. Finally, the analysis of the matrix V has also been useful because it suggested possible similarities among the attributes.

The experiments on data sets with a high number of attributes demonstrate that the number of groups of attributes shown through the 2-D visualization of V, approximately corresponds to the number of clusters found in the data set.

In conclusion, this work discussed a practical data mining methodology based on the SVD technique for discovering clusters when no predefined knowledge about data classes is available. The main goal of our work was to examine and evaluate SVD on real data sets to assess the technique. The results on the two data sets described here and on other data sets suggest that SVD is able to find clusters and represent them as groups of items in a d-dimensional space ($d \leq 3$) when these are separable in the original m-dimensional space. A user, by interactively inspecting the d-dimensional graphical representation, can formulate hypotheses by separating the groups he deems to be present in the data set on the base of space density. A tentative assignments of points to the identified clusters is done by defining threshold values on the projected attributes. To complete the data mining process and assess the validity of the hypothesis, a description of the conjectured model through decision trees or production rules can be built by running a classification algorithm that uses, as class attributes, those derived by SVD.

References

[Bel73] E. Beltrami. Sulle funzioni bilineari [on bilinear functions]. *Giornale di Matematiche ad Uso degli Studenti delle Università*, 11:98–106, 1873.

[BM] C.L. Blake and C.J. Merz. Uci repository of machine learning. In *http://www.ics.uci/mlearn/MLRepository.html*, Irvine, CA: University of California, Department of Information and Computer Science.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[Eve77] B. Everitt. *Cluster Analysis.* Heinemann Educational Books Ltd, London, 1977.

[FS96] Piatesky-Shapiro G. Fayyad, U.M. and P. Smith. From data mining to knowledge discovery: an overview. In *Advances* *in Knowledge Discovery and Data Mining*, pages 1–34, In U.M. Fayyad & al. (Eds), AAAI/MIT Press, 1996.

[GVL89] G.H. Golub and C.F. Van Loan. *Matrix Computation.* The Johns Hopkins University Press, 1989.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[Jol86] I.T. Jolliffe. *Principal Component Analysis.* Springer Verlag, 1986.

[Kor97] Jagadish-H.V. & Faloutsos C. Korn, F. Efficient supporting ad hoc queries in large datasets of time sequences. In *Proceedings of the ACM Sigmod Conf. on Magagment of Data*, pages 289–300, ACM Press, 1997.

[KR90] L. Kaufman and P.J. Rousseew. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley and Sons, 1990.

[Pre92] Teukolsky S.A. Vetterling W.T. & Flannery B.P. Press, W. H. *Numerical Receips in C: The Art of Computing.* Cambridge University Press, 1992.

[Str80] G. Strang. *Linear Algebra and its Applications.* Academic Press, 1980.

Chapter 24

SPATIAL DEPENDENCE IN DATA MINING

James P. LeSage and R. Kelley Pace

Abstract Data sets that represent observational units that are located at different points on a map often exhibit spatial dependence. Symptoms of such spatial dependence include clustering of similar values by location (e.g., house prices and incomes are similar in a subdivision, wetlands usually lie near other wetlands, pollution gradually dissapates away from the source). This chapter discusses alternative models and methods that can be used to estimate regression relationships for this type of sample data.

Keywords: Maximum likelihood estimation, matrix exponential specification, log determinant.

1. Introduction

Recent advances in address matching of customer information, house addresses, retail locations, and automated data collection using GPS systems have produced very large datasets that exhibit spatial dependence. We first motivate the need to carefully consider spatial aspects of such data with a comparison of spatial and non-spatial models based on a 1999 consumer expenditure survey and the 1990 Census. The consumer expenditure survey collects data on categories such as alcohol, tobacco, food, and entertainment in over 50,000 census tracts. To predict each of the 51 subcategories as well as overall consumer expenditures by census tract, we used 25 explanatory variables including typical variables such as age, race, gender, income, age of homes, house prices, and so forth. The predictive performance of a simple least-squares model exhibited a median absolute percentage error of 8.21% across the categories (minimum 7.69% for furnishing expenditures and maximum of 8.91% for personal insurance expenditures) and the median R^2 was 0.94 across all categories. For comparison, a more complex model using the variables and their squared values, along with these variables interacted with latitude and longitude, as well as latitude interacted with longitude, latitude squared, longitude squared, and

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 439–460.
© 2001 *Kluwer Academic Publishers.*

50 state dichotomous variables was constructed, producing a total of 306 variables. This specification provides a way of incorporating both functional form and spatial information, allowing each parameter to have a quadratic surface over space. In addition, the constant term in this model can follow a quadratic over space combined with a 50 state step function. As expected the addition of these 281 extra variables improved performance of the model, producing a median absolute percentage error of 6.99% across all categories. Both of these approaches yield reasonable results and one might wonder whether more elaborate modeling could substantially reduce the errors.

To address this, we estimated via maximum likelihood a spatial model with the original 25 explanatory variables plus spatial explanatory variables based on a weighted average of 30 nearest tracts. This simple model based on 54 variables resulted in a median absolute percentage error of 6.23% across all categories, which clearly outperformed the complicated model with 306 variables. The spatial autoregressive coefficient across the categories varied from a low of 0.72 to a high of 0.74, pointing to strong spatial dependence in the sample data. Note, the model subsumes the usual OLS model and so testing for the presence of spatial autocorrelation is the equivalent to a likelihood ratio test for the autoregressive parameter equal to 0. For these data such likelihood ratio tests overwhelmingly reject spatial independence. Such coherent integration of testing and estimation is a great advantage of likelihood-based approaches.

Relative to the non-spatial least-squares model based on 26 variables, the spatial model dramatically reduced prediction errors from 8.21% to 6.23%. These results are representative of our experience that a conventional regression augmented with geographic dichotomous variables or parameters constructed to vary over space can rarely outperform a simpler spatial model. Anyone seriously interested in prediction of georeferenced data should consider a spatial model.

Computational considerations have traditionally prevented application of spatial models to large datamining applications, but we discuss a number of recent advances that allow use of these models in datamining. We focus on computational issues related to likelihood-based methods that provide the basis for both maximum likelihood and Bayesian estimation. Reliance on the likelihood function avoids the problem of overfitting often encountered in ad-hoc procedures used in datamining and provides a sound basis for inference.

Section 2 sets forth the nature of spatial dependence along with common approaches to incorporating this in statistical modeling relationships. Section 3 discusses computational problems that arise in conventional estimation of these models with large sample sizes typically encountered in datamining applications. This section also incorporates recent advances that allow rapid estimation of spatially dependent relationships for large data samples. A series of functions written in the MATLAB and Fortran 90 languages are available to carry out the estimation procedures described here at: www.spatial-statistics.com and www.spatial-econometrics.com. Throughout the presentation we provide computational speed comparisons of the various methods and in section 4 we provide an application that compares estimates and computational times.

2. Modeling Spatial Dependence

Spatial dependence in a collection of sample data implies that observations at location i depend on other observations at locations $j \neq i$. Formally, we might state:

$$y_i = f(y_j), i = 1, \ldots, n \quad j \neq i \tag{24.1}$$

Note that we allow the dependence to be among several observations, as the index i can take on any value from $i = 1, \ldots, n$. Spatial dependence can arise from theoretical as well as statistical considerations. From a theoretical viewpoint, consumers in a neighborhood may emulate each other leading to spatial dependence. Local governments might engage in competition that leads to local uniformity in taxes and services. Pollution can create systematic patterns over space, and clusters of consumers who travel to a more distant store to avoid a high crime zone would also generate these patterns. From a statistical perspective, spatial dependence can arise from unobservable latent variables that are spatially correlated. Our motivating example illustrated that consumer expenditures exhibit spatial dependence among census tracts as do other variables such as housing prices. It seems plausible that difficult-to-quantify or unobservable characteristics such as the quality of life may also exhibit spatial dependence. In many data mining applications, the spatial structure of the dependence may be a subject of interest or provide a key insight. It may also be the case the inappropriate treatment of sample data with spatial dependence could obscure important information or insights.

There are several ways to quantify the structure of spatial dependence between observations, but a common specification relies on an nxn spatial weight matrix D with elements $D_{ij} > 0$ for observations $j = 1 \ldots n$ sufficiently close (as measured by some metric) to observation i. Note that we require knowledge of the location associated with the observational units to determine the closeness of individual observations to other observations. Sample data that contains address labels could be used in conjunction with Geographical Information Systems (GIS) or other dedicated software to measure the spatial proximity of observations. Assuming that address matching or other methods have been used to produce x-y coordinates of location for each observation in Cartesian space, we can rely on a Delaunay triangularization scheme to find neighboring observations. To illustrate this approach, Figure 1 shows a Delaunay triangularization centered on an observation located at point A. The space is partitioned into triangles such that there are no points in the interior of the circumscribed circle of any triangle. Neighbors could be specified using Delaunay contiguity defined as two points being a vertex of the same triangle. The neighboring observations to point A that could be used to construct a spatial weight matrix are B, C, D, E, F.

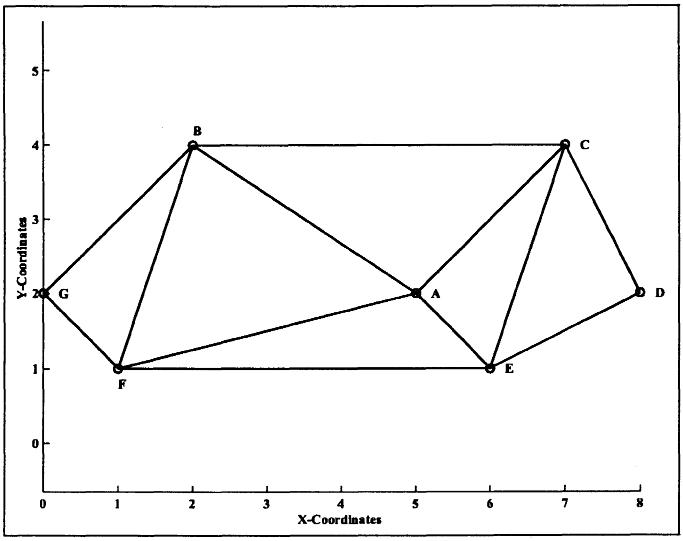


Figure 1. Delaunay triangularization

One way to specify a spatial weight matrix would be to set column elements associated with neighboring observations B, C, E, F equal to 1 in row A. This would reflect that these observations are neighbors to observation A. Typically, the weight matrix is standardized so that row sums equal unity, producing a row-stochastic weight matrix. Row-stochastic spatial weight matrices, or multidimensional linear filters, have a long history of application in spatial statistics (e.g., [Ord75]).

An alternative approach would be to rely on neighbors ranked by distance from observation A. We can simply compute the distance from A all other observations and rank these by size. In the case of figure 1 we would have: a nearest neighbor E, the nearest 2 neighbors E, C, nearest 3 neighbors E, C, D, and so on. Again, we could set elements $D_{Aj} = 1$ for observations j in row A to reflect any number of nearest neighbors to observation A, and transform to row-stochastic form. This approach might involve including a determination of the appropriate number of neighbors as part of the model estimation problem. A computational point here is that the use of nearest neighbors can accelerate computation because the indices to neighboring observations can be placed in an nxm matrix G where $G_{1...m}$ represent a set of m individual index vectors associated with m neighbors. For a vector y the values of $N_1 y$ for observation 1 equal $y(G_1)$, and a similar result holds for all other observations. Using index arithmetic in place of matrix multiplication can greatly reduce computation time as indexing into a matrix is one of the faster digital operations. From our earlier discussion of operations counts needed to compute Dy via sparse matrix multiplication, the nearest neighbor spatial weight matrix approach with m non-zero entries in each row results in an operation count $O(mn)$ that is linear in n.

The spatial weight matrix can be used to specify dependencies among the observational units in the nx1 vector y as shown in (24.2).

$$y = \alpha Dy + \varepsilon \tag{24.2}$$

In (24.2) we set $D_{ii} = 0$ to preclude an observation from directly predicting itself. If we rely on nearest neighbors as the metric for closeness, we can set $D_{ij} > 0$ for the nearest neighbors of observation i, and $D_{ij} = 0$ otherwise. This is convenient because $D^2_{ij} > 0$ contains neighbors to these nearest neighbors for observation i and similar relations hold for higher powers of D that identify higher-order neighbors. The scalar parameter α in (24.2) reflects the strength of spatial dependence with values of zero indicating no dependence and values near unity reflecting high dependence. The nx1 vector ε reflects "noise" or a stochastic disturbance in the spatial dependence relation. We typically assume normal iid errors for ε, but the likelihood approach developed here easily extends to other continuous densities such as the Laplace (associated with minimizing absolute errors). Statistical tests for spatial dependence could be carried out by estimating the parameter α and testing the null hypothesis $\alpha = 0$. Estimation of parameters such as α is the subject of the next section, but here we simply note that least-squares regression estimates of α will be biased and inconsistent.

While D is a nxn matrix, in typical problems D will be sparse. For a spatial weight matrix based upon Delaunay triangles among the n points in two-dimensions, the average number of neighbors for each observation will equal 6 and so the matrix will have $6n$ non-zeros out of n^2 possible elements, leading to $(6/n)$ as the proportion of non-zeros. Matrix multiplication and the various matrix decompositions require $O(n^3)$ operations for dense matrices, but for sparse D these operation counts can fall as low as $O(n_{\neq 0})$, where $n_{\neq 0}$ denotes the number of non-zeros.

Returning to our example, we can add explanatory variables to the spatial dependence relationship in (24.2) producing a regression relationship shown in (24.3), where the nxk matrix X contains explanatory variables. The parameters β might be of interest as they measure the marginal impact of the various characteristics on the dependent variable, and interest might also center on predictions of y.

$$y = \alpha D y + X\beta + \varepsilon \tag{24.3}$$

We can extend the model in (24.3) to a *mixed model* that allows for explanatory variables from neighboring observations, as shown in (24.4).

$$y = \alpha D y + X\beta + DX\gamma + \varepsilon \tag{24.4}$$

The kx1 parameter vector γ measures the marginal impact of the explanatory variables from neighboring observations on the dependent variable y. In terms of consumer spending γ might reflect the impact of the spending patterns of neighbors on the same street on an individual's buying decisions.

Maximum likelihood estimation of the model in (24.3) (and (24.4) as well as a host of related variants on these models described in [Ans88]) involves maximizing the log likelihood function (concentrated with respect to σ^2, the noise variance associated with ε) in (24.5) with respect to the parameters α and β.

$$\begin{aligned}
\ln L &= \kappa + \ln|I_n - \alpha D| - (n/2)\ln(e'e) \\
e &= e_o - \alpha e_d \\
e_o &= y - X\beta_o \\
e_d &= Dy - X\beta_d \\
\beta_o &= (X'X)^{-1}X'y \\
\beta_d &= (X'X)^{-1}X'Dy
\end{aligned} \tag{24.5}$$

Where κ represents a constant not involving the parameters. The computationally troublesome aspect of this is the need to compute the log-determinant of the nxn matrix $(I_n - \alpha D)$. Operation counts for computing this determinant grow with the cube of n for dense matrices.

3. Approaches to Estimation

Naive use of likelihood-based estimation would require $O(n^3)$ operations, which would preclude its use in large-sample data mining applications. One alternative is to devise new approximate estimators based on asymptotic statistical theory, but large-sample statistical approximations typically used to motivate these estimators in non-spatial situations may not work with spatial data. It is usually the case that straightforward minimization of some criterion function tends to overfit the spatial parts of the model arising from simultaneous dependence among observations. Standard simplifications often rely on sample independence which is clearly inappropriate, and inference regarding estimates arising from newly proposed estimators is often difficult.

We approach the problem from a numerical standpoint, devising methods that allow likelihood-based estimation for large data sets and models in (24.3) or (24.4). This allows users to rely on traditional statistical knowledge concerning these techniques as well as extensions to non-standard situations involving missing values, censored or truncated dependent variables, non-normal distributions, and flexible functional forms. Many of these non-standard situations can be flexibly handled using Bayesian approaches involving Markov Chain Monte Carlo (MCMC) methods as illustrated in [LeS97], where robust methods were applied in the presence of outliers. In addition to outliers often encountered in data mining applications, models based on binary or censored dependent variables can be easily estimated using MCMC methods demonstrated in [LeS00]. However, this estimation approach requires that samples be drawn from the conditional distribution of each parameter in the model, and the conditional distribution for the spatial dependence parameter α contains the log-determinant term. Following [PB97], [LeS00] proposed computing and storing log-determinant values over a grid of values for the parameter α. These were used with a table look-up scheme when evaluating the conditional distribution for α to avoid calculating the log-determinant on each draw from the conditional distribution for α. The techniques we present here to accelerate computation of the log-determinant term will aid Bayesian computations as well as maximum likelihood methods.

We set forth alternative approaches to likelihood-based estimation that dramatically improve the computational time and storage requirements for models with spatial dependence. The alternative approaches described here avoid direct computation of the log-determinant in the likelihood, thereby improving estimation speed and storage requirements. Specifically, we compare the computational aspects of: a direct sparse matrix approach to log-determinant computation, a Monte Carlo estimator for the log-determinant, an approach based on an alternative spatial specification that relies on a matrix exponential transform to eliminate the log-determinant from the likelihood, a method that exploits analytically derived upper and lower bounds on the log-determinant term in the likelihood function, and a divide and conquer strategy based on partitioning of the sample data along with bounds on the change in the log-determinant from ignoring some spatial interaction among partitions.

3.1. Using Sparse matrix algorithms

While the log-determinant term may seem a nuisance, in many ways its presence simplifies spatial modeling. The likelihood without the log-determinant term produces a least-squares criterion. Minimizing this criterion results in a spatial parameter that is too large, a situation akin to overfitting often encountered in datamining. Attempting to prevent this overfitting with techniques such as cross-validation often require more computing time than required to calculate the log-determinant term. In addition, the implicit penalties assigned by other methods may not provide the correct penalty, resulting in overfitting or underfitting of the spatial parameters.

One of the earlier improvements was proposed by [PB97] who suggested the use of direct sparse matrix algorithms such as the Cholesky or LU decompositions to compute the log-determinant over a grid of values for the parameter α restricted to the interval $(-1/\lambda_{min}, 1)$, where λ_{min} represents the minimum eigenvalue of the spatial weight matrix. However, since negative spatial autocorrelation is of little interest in many cases, by restricting α in the interval $[0, 1)$ one can accelerate the computations. This along with a vector evaluation of the log-likelihood function over this grid of log-determinant values was used to find maximum likelihood estimates. Specifically, they show that for a grid of q values of α in the interval $[0, 1)$,

$$\begin{pmatrix} \mathrm{Ln}L(\beta, \alpha_1) \\ \mathrm{Ln}L(\beta, \alpha_2) \\ \vdots \\ \mathrm{Ln}L(\beta, \alpha_q) \end{pmatrix} \propto \begin{pmatrix} \mathrm{Ln}|I_n - \alpha_1 D| \\ \mathrm{Ln}|I_n - \alpha_2 D| \\ \vdots \\ \mathrm{Ln}|I_n - \alpha_q D| \end{pmatrix} - \frac{n}{2} \begin{pmatrix} \mathrm{Ln}(\phi(\alpha_1)) \\ \mathrm{Ln}(\phi(\alpha_2)) \\ \vdots \\ \mathrm{Ln}(\phi(\alpha_q)) \end{pmatrix} \tag{24.6}$$

where $\phi(\alpha_i) = e_o' e_o - 2\alpha_i e_d' e_o + \alpha_i^2 e_d' e_d$. The computationally intense part of this approach is still calculating the log-determinant, which takes around 201 seconds for a sample of 57,647 observations representing all Census tracts in the continental US using a grid of 100 values from $\alpha = 0$ to 1 based on the sparse matrix algorithms in MATLAB version 6.0 on a 600 Mhz Pentium III computer. Note, if the optimum α occurs on the boundary (i.e., 0), this indicates the need to consider negative values of α.

In applied settings, one may not require the precision of the direct method and so it seems worthwhile to examine other approaches to the problem that simply approximate the log-determinant.

3.2. Log-determinant Estimator

An improvement based on a Monte Carlo estimator for the log determinant suggested by [BP99] allows larger problems to be tackled without the memory requirements or sensitivity to orderings associated with the direct approach. The estimator for $\ln|I_n - \alpha D|$ is based on an asymptotic 95% confidence interval, $(\bar{V} - F, \bar{V} + F)$ for the log-determinant constructed using the mean $\bar{V}$ of p generated independent random variables taking the form:

$$V_i = -n \sum_{k=1}^{m} \frac{x_i' D^k x_i}{x_i' x_i} \frac{\alpha^k}{k}, \quad i = 1, \ldots, p \tag{24.7}$$

where $x_i \sim N(0,1)$, x_i independent of x_j if $i \neq j$. This is just a Taylor series expansion of the log-determinant that relies on the random variates x_i to compute the trace without multiplying n by n matrices. Note that $tr(D) = 0$ and we can easily compute $tr(D^2) = \sum\sum D \odot D'$ using elementwise multiplication represented by $\odot$. In the case of symmetric D this reduces to taking the sum of squares of all the elements. This requires a number of operations proportional to the non-zero elements which are small since D is a sparse matrix. By replacing the estimated traces with exact ones, the precision of the method can be improved at low cost. In addition,

$$F = \frac{n\alpha^{m+1}}{(m+1)(1-\alpha)} + 1.96\sqrt{s^2(V_1, \ldots, V_p)/p} \tag{24.8}$$

where s^2 is the estimated variance of the generated V_i. Values for m and p are chosen by the user to provide the desired accuracy for the approximation. The method provides not only an estimate of the log-determinant term, but an empirical measure of accuracy as well in the form of upper and lower 95% confidence interval estimates. The log-determinant estimator yields the correct expected value for both symmetric and asymmetric spatial weight matrices, but the confidence intervals above require symmetry. For asymmetric matrices, a Chebyshev interval may prove more appropriate.

An alternative approach to assessing the quality of the estimate for the log-determinant can be based on an examination of estimates for α derived from each realization i. This allows an examination of the variation in the model estimates that result from using an estimated log-determinant in place of the exact result. We discuss results from this approach below.

The computational properties of this estimator are related to the number of non-zero entries in D which we denote as f. Each realization of V_i takes time proportional to fm, and computing the entire estimator takes time proportional to fmp. For cases where the number of non-zero entries in the sparse matrix D increase linearly with n, the required time will be order nmp.

In addition to the time advantages of this algorithm, memory requirements are quite frugal because the algorithm only requires storage of the sparse matrix D along with space for two nx1 vectors, and some additional space for storing intermediate scalar values. Additionally, the algorithm does not suffer from "fill-in" since the storage required does not increase as computations are performed.

Another advantage is that this approach works in conjunction with the calculations proposed by [PB97] for a grid of values over α, since a minor modification of the algorithm can be used to generate log-determinants for a simultaneous set of α values $\alpha_1, \ldots, \alpha_q$, with very little additional time or memory. To see this, note that for each realization of V_i the vector $(x_i' D x_i, \ldots, x_i' D^k x_i)$ needs to be computed, but this can then be multiplied by a kxq matrix with i, jth entry equal to α_i^j / j. Since the computational effort is mainly centered on the quadratic forms, additional values for α can be evaluated without appreciably increasing the time or memory requirements of the algorithm. This grid can extend over negative or positive values (less than one).

Finally, the Monte Carlo estimator would be a candidate for parallel computation since different processors could simultaneously determine estimators V_i independently with no need to communicate. These computed V_i values could then be combined when the computations were finished and no modification of the remaining code would be required.

As an illustration of these computational advantages, the time required to compute a grid of log-determinant values for $\alpha = 0, \ldots, 1$ based on 0.001 increments for the sample of 57,647 observations was 3.6 seconds when setting $m = 20, p = 5$, which compares quite favorably to 201 seconds for the direct sparse matrix computations cited earlier. This approach yielded nearly the same estimate of α as the direct method (0.91 versus 0.92), despite the use of a spatial weight matrix based on pure nearest neighbors rather than the symmetricized nearest neighbor matrix used in the direct approach. To further illustrate the utility of the log-determinant estimator, we generated 1,000 iterations ($p = 1,000$) of the MC log-determinant estimator (which took 631.2 seconds or 0.63 seconds per iteration) and computed maximum likelihood estimates based on each log-determinant curve produced by a single iteration. Of the 1,000 maximum likelihood estimates for α, the minimum and maximum estimates were 0.905 and 0.915, with a standard deviation of 0.001543. The distribution of these estimates appeared approximately normal with a studentized range under 6.5. The robustness of the simple Monte Carlo log-determinant estimator is rather remarkable and suggests a potential for other approximate approaches.

3.3. A Matrix Exponential Specification

In [LP00] an estimation approach that leads to closed-form maximum likelihood estimates is presented. They introduce the matrix exponential spatial specification (MESS) that relies on a matrix exponential spatial transformation of the dependent variable and show that the troublesome log determinant term vanishes from the log likelihood function in this specification. Specifically, models with spatial dependence can be viewed as representing a situation where the dependent variable y undergoes a linear transformation Sy as in (24.9).

$$Sy = X\beta + \varepsilon \tag{24.9}$$

Where the matrix S represents a positive definite nxn matrix resulting in a log-likelihood for the MESS model shown in (24.10), where C represents a scalar constant and both $M = I - H$ and $H = X(X'X)^{-1}X'$ are idempotent matrices. The term $|S|$ is the Jacobian of the transformation from y to Sy.

$$L = C + \ln|S| - (n/2)\ln(y'S'MSy) \tag{24.10}$$

The matrix exponential defined in (24.11) can be used to model S as:

$$S = e^{\alpha D} = \sum_{i=0}^{\infty} \frac{\alpha^i D^i}{i!} \tag{24.11}$$

For the nearest neighbor specification of D, the matrix exponential S can be interpreted as assigning rapidly declining weights for observations involving higher-order neighboring relationships. That is, observations reflecting higher-order neighbors (neighbors of neighbors) receive less weight than lower-order neighbors.

A property of the matrix exponential, $|e^{\alpha D}| = e^{\text{trace}(\alpha D)}$ allows us to simplify the MESS log-likelihood. Since trace$(D) = 0$ and by extension $|e^{\alpha D}| = e^{\text{trace}(\alpha D)} = e^0 = 1$, the log-likelihood takes the form: $L = C - (n/2)\ln(y'S'MSy)$. This produces a situation where maximizing the log-likelihood is equivalent to minimizing $(y'S'MSy)$ with respect to S. This is essentially a closed-form problem in that it involves solving a constrained non-linear (polynomial) least-squares problem.

The MESS model in (24.9) can be extended to allow for spatial dependence in the explanatory variables matrix X as shown in (24.12).

$$X = [\iota\ U\ DU \ldots D^{q-1}U] \tag{24.12}$$

Where U represents a matrix of observations on p non-constant independent variables and q is an integer large enough so that X approximately spans SU, but small enough so that X cannot span y.

Additional flexibility can be introduced by including a spatial decay parameter ρ that lies between 0 and 1 in the spatial weight specification based on nearest neighbors. If we let N_i represent a matrix containing the i nearest neighbors, the flexible weight structure is shown in (24.13), where m denotes the maximum number of neighbors considered.

$$D = \frac{\sum_{i=1}^{m} \rho^i N_i}{\sum_{i=1}^{m} \rho^i} \tag{24.13}$$

In (24.13), ρ^i weights the relative effect of the ith individual neighbor matrix, so that S depends on the parameters ρ as well as m in both its construction and the metric used. By construction, each row in N sums to 1 and has zeros on the diagonal. To see the role of the spatial decay parameter ρ, consider that a value of $\rho = 0.87$ implies a decay profile where the sixth nearest neighbor exerts less than 1/2 the influence of the nearest neighbor. We might think of this value of ρ as having a "half-life" of six neighbors. On the other hand, a value of $\rho = 0.95$ has a half-life between 14 and 15 neighbors.

As already noted, row-stochastic, non-negative D can have a maximum of 1 in any row, so the magnitude of the elements in the powers of D does not grow with the power, suggesting that the power series in (24.11) converges rapidly. Given the rapid decline in the coefficients of the power series, [LP00] report achieving a satisfactory progression with six or seven terms. Computationally, we do not rely on a straightforward implementation of the definition of S by a power series expansion truncated after q terms. This would be impractical as it would require $O((q-2)n^3)$ operations for dense D, because dense matrix multiplication requires $O(n^3)$ operations. Fortunately, one does not need to compute S separately, as S always appears in conjunction with y which allows computation of Sy in $O((q-1)n^2)$ operations for dense D by sequential left-multiplication of y by D to form $n-$element vectors, (i.e., $Dy, D(Dy) = D^2y$, and so on). For sparse D, the number of operations required drops to $O((q-1)n_{\neq 0})$, where $n_{\neq 0}$ denotes the number of non-zeros. For the nearest neighbor spatial weight matrix approach with m non-zero entries in each row, the operation count associated with computing Sy would decline to $O((q-1)mn)$. This results in an operation count for computing Sy in nearest neighbor specifications of D that is linear in n.

To illustrate this computation approach in detail, we define the nxq matrix Y comprised of powers of D times y in (24.14).

$$Y = [y \; Dy \; D^2y \ldots D^{q-1}y] \tag{24.14}$$

Note that, this simple form of sparse matrix-vector multiplication can be implemented without explicit use of sparse matrix multiplication algorithms. Given a list of m neighbors for observation i (i.e., $(i, j_1), (i, j_2), \ldots, (i, j_m)$) and given equal weights for the neighbors, the product Dy for observation i is merely $(y_{j_1} + y_{j_2} + \ldots + y_{j_m})/m$. That is, the sparse matrix-vector products needed to compute Y in (24.14) require only indexing and addition, two of the fastest operations possible on digital computers. This means that the MESS model can be implemented in a variety of computing languages such as Fortran 90 or C, and various statistical software environments. In fact, to demonstrate the feasibility of estimating MESS using standard software we coded the estimator in Fortran 90 as well as in Matlab. Employing a compiled language (Fortran 90) plus using indexing as opposed to sparse matrices increased the speed by more than 6 times.

As an illustration of the speed associated with this approach, MESS estimates for a model based on our sample of 57,647 observations took less than 0.5 seconds on a 600 Mhz desktop PC.

3.4. Double Bounded Estimates (DBLE)

The approach described here avoids computing the log-determinant by replacing it with upper and lower bounds on this term. These bounds are easy to compute, and the profile log-likelihood for both the lower and upper bounds has a closed-form solution based on a simple cubic equation in α. This in conjunction with the fact that maximum likelihood estimates for α lie between those at the bounds may allow inferences based on the log-likelihood at the bounds, avoiding the need to compute the actual log-likelihood. Intuitively, this type of result would arise if one would draw the same inference based on the bounded log-likelihoods as with the actual log-likelihoods. [PW92] discuss this issue and provide the label *likelihood dominance* to the type of reasoning applied here.

To illustrate how likelihood dominance can lead to valid inferences based on the bounds, consider a case where the focus of inference is on J restrictions imposed on a spatial model. Likelihood ratio tests based on conventional maximum likelihood estimates would rely on twice the difference between the restricted and unrestricted likelihoods which follow an asymptotic $\chi^2(J)$ distribution. Suppose the unrestricted model likelihood lies in the interval [-1200 -1000] and the restricted model likelihood lies in the interval [-1500 -1300]. Since the intervals do not come close to overlapping, neither do the actual likelihoods, allowing us to reject the restrictions (provided these are not too numerous) without computing the exact likelihood.

In *Proposition 1*, we provide bounds on the log-determinant. The proof of this is in the appendix.

Proposition 1. For a symmetric nonnegative matrix D with eigenvalues $\lambda_{\min} \geq -1$, $\lambda_{\max} \leq 1$, $(1/\lambda_{\min}) < \alpha < 1$, and $tr(D) = 0$,

$$(\alpha + \ln(1-\alpha)) \sum\sum D_{ij}^2 \leq \ln|I - \alpha D| \leq -\frac{\alpha^2}{2} \sum\sum D_{ij}^2 \qquad (24.15)$$

Given symmetry, $\sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}^2$ is computationally feasible as it requires only sums of squared elements from the upper (or lower) triangle which can be multiplied by 2. In spatial applications where D is sparse and the number of non-zeros per row do not grow with n, this calculation would require only $O(n)$ operations.

Finally, a closed-form expression for estimates of α at the bounds which we denote α can be found using the profile log-likelihood. This requires that we formulate the profile log-likelihood shown in (24.16) where: $M = I_n - X(X'X)^{-1}X'$, $\hat{e}_y = My$ the least-squares residuals from fitting y to X, $\hat{e}_{Dy} = MDy$ the residuals from fitting Dy to X, $\alpha_g = (e'_{Dy}e_{Dy})^{-1}e'_{Dy}e_y$, $r = (e'_{Dy}e_{Dy})^{-1}e'_y e_y$, and κ is a constant not involving α. The term α_g is the upwardly biased estimate of α which results from minimizing the sum-of-squared errors without the log-determinant term.

$$L(\alpha) = \kappa + \ln|I_n - \alpha D| - (n/2)\ln(r - 2\hat{\alpha}_g\alpha + \alpha^2) \tag{24.16}$$

This also sets up the bounded log-likelihoods,

$$\begin{aligned} L^{+}(\alpha) &= \kappa - \frac{\alpha^2}{2}\sum\sum D_{ij}^2 - (n/2)\ln(r - 2\hat{\alpha}_g\alpha + \alpha^2) \qquad (24.17)\\ L^{-}(\alpha) &= \kappa + (\alpha + \ln(1-\alpha))\sum\sum D_{ij}^2 - (n/2)\ln(r - 2\hat{\alpha}_g\alpha + \alpha^2) \end{aligned}$$

One can show that the solution to the first-order conditions for a maximum of the profile log-likelihood with respect to α at the bounded log-likelihoods is a cubic equation with a closed form solution. We term this model the DBLE or double bounded likelihood estimator to emphasize this notable feature.

It took 0.4 seconds to compute $tr(D^2)$ and 2.68 seconds in total to compute the DBLE estimates for the 57,647 observation sample. The estimated lower bound was $\alpha = 0.85$ and the estimated upper bound was $\alpha = 0.97$, which bracket the exact maximum likelihood estimate $\alpha = 0.91$. Estimates at the bounds were very similar to the exact answer and since all the variables were significant at both bounds for α, one could accept the hypothesis of significance and avoid computing exact maximum likelihood estimates.

3.5. A Partitioning Scheme

Here we describe a divide and conquer strategy for spatial maximum likelihood estimation that could accelerate computation using parallel processing, or extend the size of problems handled on a single machine through the use of out-of-core execution. The strategy is based on a derivation of upper and lower bounds to the log-determinant that result from ignoring interactions between subproblems set forth in [PL00].

The intuition behind this approach is that the non-zero elements in the matrix $I_n - \alpha D_s$ cluster around the main diagonal, but are generally irreducible, so that we cannot transform this matrix to block-diagonal form via row and column permutations. The spatial weight matrix D_s is derived from an adjacency matrix N reflecting nearest neighbors. Because a nearest neighbor matrix is not symmetric, we ensure symmetry using the transformation $N_s = (N + N')/2$, and then reweight using the diagonal matrix W to produce a row-stochastic matrix $D_r = WWN_s$. This transformation destroys symmetry in that D_r is asymmetric, but we can use a similarity transformation to produce $D_s = W^{-1}D_r(W^{-1})^{-1} = WN_sW$ such that D_s and D_r will have the same eigenvalues [Ord75], p. 125. Note also that the eigenvalues of D_r do not exceed 1 in absolute value because it is a row-stochastic matrix. Since D_s and D_r are similar, they have the same eigenvalues which we label λ_i, and these eigenvalues are bounded: $-1 \leq \lambda_i \leq 1$.

A one-way dissection approach to computing the log-determinant is described in [GO93] p. 228-230, and we explore a variant of this approach that examines the impact of the separator sets arising from the one-way dissection on the log-determinant. Intuitively, the small number of non-zero terms in $I_n - \alpha D_s$ should not have a large impact on the log-determinant. We seek upper and lower bounds on the impact of ignoring these terms on the log-determinant that arise from the use of partitioned data. Using these bounds, we compute maximum likelihood estimates for the entire data set at the bounds that provide a range of parameter values over the lower to upper bound on the log-determinant. Of course, maximum likelihood estimates based on the true log-determinant would be inside this range of parameter values, and if the range is small relative to the precision required for the problem, we have a useful simplification. This particular simplification is useful because it reduces computation time and allows for multiprocessing as well as extending the size of problem that can be handled on a single processor.

The bounds on the log-determinant are shown in (24.18) where a binary indicator matrix $O_{ij} = \text{Ind}(D_{ij} \in \Omega)$ is used to select elements in the separator set Ω that one wishes to exclude. This is used to produce $U = O \odot D$, which relies on Hadamard or elementwise multiplication denoted by $\odot$ to select excluded elements.

$$\begin{aligned}
\ln|I_n - \alpha S| &\geq \ln|I_n - \alpha D| \geq \ln|(I_n - \alpha R) - \alpha S| \qquad (24.18)\\
S &= D - U\\
R_{ii} &= \sum_{j=1}^{n} U_{ij}, \quad R_{ij} = 0, \quad i \neq j\\
0 &\leq \alpha \leq 1
\end{aligned}$$

Note that the term R_{ii} represents the difference between the upper and lower log-determinant bounds which will be small if: 1) omitted elements belonging to the separator set in each row and column are small or; 2) if not many rows and columns contain separator set elements.

>From an implementation standpoint computing these upper and lower bounds can be simplified by eliminating all non-zero entries (representing interactions) in the rows and columns in the off-diagonal separator set elements. If we set these to zero in one of the off-diagonal blocks, two simplifications emerge: 1) the diagonal matrix R takes on values of 0 or 1, where $R_{qq} = 1$ indicates that q is an index of a row or column with an interaction; 2) the determinant of $(I - \alpha S_s)$ represents a block-triangular system. This simplification produces bounds shown in (24.19) based on S_s and D_s, resulting from similarity transforms that do not affect the log-determinant, but ease the computational task. In (24.19), we use o^{ω} to represent the # of interaction rows (those with index values of 1).

$$\ln|I - \alpha S_s| \geq \ln|I - \alpha D_s| \geq \ln|(I - \alpha S_s)| + o^{\omega}\ln(1 - \alpha) \qquad (24.19)$$

A proof for these bounds is given in [PL00], but here we note that the difference between the upper and lower bounds depend only on the number of rows in the off-diagonal blocks with separator set elements o^{ω}. The quality of the bounds will then depend on a good partitioning. For spatial models with inherent local structure, a natural ordering from east-west or north-south will usually provide blocks that are adequate to produce high quality bounds. However, the use of graph partitioning software such as METIS [KK00] could provide better blocks that would improve performance of the method

Computationally, note that once we compute the upper bound based on the block triangular system, $\ln|I - \alpha S_s|$, we need only a scalar computation to produce o^{ω} and the lower bound. This approach can be extended to larger samples with more blocks in a straightforward way. If an inherent local structure of spatial dependence exists, it is likely that interactions only occur between contiguous blocks, so the number of interactions neglected using this approach should be very small. To formalize this extension, let:

$$D_s = \begin{pmatrix} B_{11} & B_{12} & 0 & 0 \\ B_{21} & B_{22} & B_{23} & 0 \\ 0 & B_{32} & \ddots & B_{(k-1),k} \\ 0 & 0 & B_{k,(k-1)} & B_{kk} \end{pmatrix} \qquad (24.20)$$

then the bounds for this system can be stated as:

$$(\sum_{i=1}^{k} \ln|I-\alpha B_{ii}|) \geq \ln|I-\alpha D_s| \geq (\sum_{i=1}^{k} \ln|I-\alpha B_{ii}|+(\sum_{i=2}^{k} o^{\omega}_{i,(i-1)})) \quad (24.21)$$

Estimating the model proceeds much as it did with the previously described DBLE model, but using the set of tighter log-determinant bounds discussed above. Note, the model estimates are based on all observations and their neighbors, but the partitioning approach described here could be used to determine estimates derived from sub-samples. This would allow statistical tests for homogeneity over the sample, perhaps leading to a decision to partition the statistical model to account for heterogeneity across sample observations. Intuitively, this situation might arise if the structure of spatial dependence were different for various regions of the country, say west versus east or north versus south. Discovery of this type of systematic pattern is frequently the objective in data mining applications.

As a concrete illustration of the computational gains from this approach, [PL00] were able to compute a spatial autoregression for a sample of 3,954,400 observations in 47.8 minutes using a single processor (out-of-core execution) 600 Mhz Pentium III running Matlab 6.0. The task was divided into 50 separate blocks and so with 50 processors the time could fall to under one minute for this problem. The upper bound estimate of the spatial autoregressive parameter was 0.89 and the lower bound estimate of the spatial autoregressive parameter was 0.87. It is difficult to imagine a situation where the difference of 0.02 in an estimated parameter makes a practical difference. Note that far greater reductions in computational times would arise from the use of 1) more partitions, 2) multiple (parallel) processors and 3) optimal partitions, all of which could be applied to this problem.

4. A Comparative Illustration

Using the sample of 57,647 observations on housing values for all census tracts in the continental US we compare estimates from: two aspatial models:

1 aspatial least-squares: $y = X\beta + \varepsilon$

2 aspatial least-squares with spatial X: $y = X\beta + DX\gamma + \varepsilon$

and two spatial models that both rely on inclusion of a spatial lag of the dependent variable: $y = \alpha Dy + X\beta + DX\gamma + \varepsilon$. One of the spatial models represents a very limited spatial model since it relies on only a single nearest neighbor [PZ00]. Four explanatory variables, land area, population, per capita income, and house age were used along with spatial lags of these variables constructed using the product of the weight matrix times the variables.

Estimates are in table 1 where the first two columns represent the aspatial models and the last two columns present results for the spatial models. We see that inclusion of the spatial lag of the dependent variable produced a substantial improvement in the fit, indicated by a comparison of the log-likelihood values from these four models. In addition, inferences regarding the significance as well as marginal impacts for each of the explanatory variables would be similar for the two spatial models. In contrast, the two aspatial models suggest a larger influence of land area, population and income on housing values than the spatial models.

Table 1. Spatial and Aspatial Regression Models

Variables	Aspatial Model	Spatial X Model	1 Neighbor Model	Exact Mixed Model
ln(Land Area)	-0.085	-0.009	-0.019	-0.003
Dln(Land Area)	—	-0.113	-0.021	-0.006
Deviance	9,379.1	9,602.9	6,051.0	287.1
ln(Population)	0.115	0.018	0.039	0.023
Dln(Population)	—	0.265	0.025	-0.003
Deviance	1,358.6	2,107.1	1,115.2	224.4
ln(Income	1.084	0.640	0.722	0.676
Dln(Income)	—	0.643	-0.271	-0.546
Deviance	43,355.2	45,089.4	49,281.8	33,605.1
ln(Age)	-0.127	-0.133	-0.122	-0.140
Dln(Age)	0.000	0.064	0.074	0.127
Deviance	1,175.2	956.0	1,978.8	2,924.0
m(# of neighbors)	—	—	1	30
ρ	—	—	1	0.9
α	—	—	0.6116	0.920
Deviance($\alpha = 0$)		9,618	43,034	78,598
k(# of variables)	5	9	10	12
Log-Likelihood	-265,505	-260,696	-243,988	-226,206

To provide a comparative illustration of the time required to compute estimates using the various methods discussed as well as a comparison of estimates and inferences we present results in Table 2 from four alternative spatial methods representing: the bounded estimates (DBLE), estimates based on the Monte Carlo estimator for the log-determinant (MC), an exact model, and the matrix exponential specification (MESS). Here we see similar estimates and inferences regarding the significance and marginal impact on housing values of the four explanatory variables plus their spatial lags. The basis of comparing these methods are the log-likelihoods and the computational times. OLS provides the worst performance (most negative log-likelihood), the use of one neighbor increases the likelihood dramatically, and using 30 nearest neighbors further increases the log-likelihood.

Table 2. A Comparison of Spatial Regression Models

Variables	DBLE Lower	Exact X Model	DBLE Upper	MC Model Estimates	MESS Model
ln(Land Area)	-0.002	-0.003	-0.004	-0.004	-0.003
ln(Population)	0.022	0.023	0.023	0.023	0.022
ln(Income)	0.0674	0.676	0.678	0.682	0.677
ln(Age)	-0.140	-0.140	-0.141	-0.139	-0.138
Dln(Land Area)	-0.013	-0.006	-0.000	-0.010	-0.017
Dln(Population)	0.015	-0.003	-0.017	0.002	0.030
Dln(Income)	-0.468	-0.546	-0.612	-0.537	-0.463
Dln(Age)	0.12	0.127	0.131	0.127	0.127
α	0.860	0.920	0.970	0.910	0.812
Log-Like	-227,890	-226,206	-224,608	-226,217	-228,851
Weight Matrix	sym	sym	sym	asym	asym
Total time[a]	1.34	201.7	1.34	4.5	0.5
ln(det) time	0.4	200.8	0.4	3.6	

[a] Times are in seconds, for MATLAB in the DBLE and Exact, FORTRAN90 in the MC and MESS.

The time required to produce estimates using the four alternative estimation methods varies greatly from a high of 201.7 seconds for sparse computation of exact maximum likelihood estimates to a low of 0.5 seconds for the MESS model. We used the weighted nearest neighbors because they produced much higher log-likelihoods than using Delaunay weights. Use of the Delaunay weights or fewer neighbors would further accelerate the computational times. Extrapolating, some of these methods should continue to work well with very large data sets.

5. Summary

Our presentation shows that maximum likelihood estimates for models involving sample data that exhibit spatial dependence is feasible for large samples typically encountered in datamining applications. The virtues of likelihood-based estimates versus ad-hoc or least-squares estimates are numerous. First, inclusion of a single spatial lag of the dependent variable provides an improvement in fit as well as dramatic differences in inferences drawn concerning the significance and impact of explanatory variables. Second, likelihood estimation of the spatial autoregression provides a coherent test for the presence of spatial dependence as a byproduct (test for the autoregressive parameter equal to 0). Third, since datamining applications frequently involve a search for meaningful relationships in the sample data which often has spatial dimensions, appropriate treatment of spatial dependence critically affects the ability to find hidden relations in the data. As a motivating example of this we demonstrated that a mixed regressive spatially autoregressive model of consumer expenditures with

54 variables produced substantially lower prediction errors than a least-squares model containing 306 explanatory variables.

One can further refine spatial techniques by incorporating a number of adjustments. For example, simultaneously estimating the functional form and spatial dependence structure may result in a higher estimate of the spatial parameter and even better predictive performance ([Pacng]).

In terms of a typical modeling strategy, we suggest using MESS to explore potential relations among the data using a sample from a geographically compact region extracted from the overall data and fitting a final model either via MESS or via a standard spatial model with an estimated log-determinant. Aspatial random sampling is often used in datamining, but this operation destroys spatial relations and hence forfeits the many advantages of recovering information by examining the errors or values from nearby operations. Hence, the admonition to use a sample from a compact region (e.g., all census tracts in a state, all individuals in a city, etc).

Although our focus here was on two-dimensional space, data with both spatial and temporal attributes can exhibit spatial-temporal dependence (e.g., [GS00]) and one can imagine spatial relations that take place in higher dimensions as well. Some of the approaches set forth here extend to multiple dimensions where identifying and estimating patterns of multidimensional dependence could provide the types of insight sought in datamining.

Appendix

Proof. Consider the Taylor series expansion:

$$\ln|I - \alpha D| = -\sum_{i=1}^{\infty} tr(D^i)\frac{\alpha^i}{i} \tag{A.1}$$

Inspection of this shows $-\sum_{i=1}^{\infty} tr(D^i)\left(\frac{\alpha^i}{i}\right) \leq -\frac{\alpha^2}{2}tr(D^2)$ since $-\frac{\alpha^2}{2}tr(D^2)$ equals the first two terms of the series because $tr(D) = 0$, while $tr(D^i) \geq 0$ for $i = 3...\infty$, since D is nonnegative. Hence,

$$\ln|I - \alpha D| \leq -\frac{\alpha^2}{2}tr(D^2) \tag{A.2}$$

and because $tr(D^2) = \sum\sum D_{ij}^2$ by virtue of symmetry, this proves the upper bound.

For the lower bound, symmetry of D implies real eigenvalues λ_i such that $|\lambda_i| \leq 1$ for $i = 1, ..., n$. Even powers of the matrix have all nonnegative eigenvalues since $\lambda_i^{2j} \geq 0$ for positive integer j and via properties of the trace, $tr(D^{2j}) = \sum_{i=1}^{n} \lambda_i^2$. Furthermore, for positive integer r:

$$tr(D^{2j+r}) = \sum_{i=1}^{n} \left(\lambda_i^2\right)\lambda_i^r \tag{A.3}$$

Since the maximum value for λ_i^r is 1 for any positive integer r, $\lambda_i^{2j} \geq 0$, and $tr(D) = 0$,

$$\begin{array}{rcl} tr(D^{2j}) & \geq & tr(D^{2j+r}) \text{ and } \textit{a fortiori} \\ tr(D^2) & \geq & tr(D^{2+r}) \\ \ln|I - \alpha D| & = & -\sum_{i=2}^{\infty} tr(D^i)\frac{\alpha^i}{i} \\ -\sum_{i=2}^{\infty} tr(D^i)\frac{\alpha^i}{i} & \geq & -tr(D^2)\sum_{i=2}^{\infty}\frac{\alpha^i}{i} \end{array}$$

Moreover,

$$-\sum_{i=2}^{\infty}\frac{\alpha^i}{i} = \left(\alpha + -\sum_{i=1}^{\infty}\frac{\alpha^i}{i}\right) \tag{A.4}$$

which equals $(\alpha + \ln(1 - \alpha))$. Hence,

$$\ln|I - \alpha D| \geq tr(D^2)\left(\alpha + \ln(1 - \alpha)\right) \tag{A.5}$$

and since $tr(D^2) = \sum\sum D_{ij}^2$ by virtue of symmetry this proves the lower bound. QED.

References

[Ans88] Luc Anselin. *Spatial Econometrics: Models and Methods*. Kluwer Academic Publishers, Dorddrecht, 1988.

[BP99] R. Barry and R. Kelley Pace. A monte carlo estimator of the log determinant of large sparse matrices. *Linear Algebra and its Applications*, 289:41–54, 1999.

[GO93] G. H. Golub and J. M. Ortega. *Scientific Computing: An Introduction with Parallel Computing*. Academic Press, Inc., 1993.

[GS00] R. Kelley Pace R. Barry O.W. Gilley and C.F. Sirmans. A method for spatial-temporal forecasting with an application to real estate prices. *International Journal of Forecasting*, 16:229–246, 2000.

[KK00] K. Schloegel G. Karypis and V. Kumar. Graph partitioning for high performance scientific simulations. In J. Dongarra et al, editor, *CRPC Parallel Computing Handbook*. Morgan Kaufman, 2000.

[LeS97] James P. LeSage. Bayesian estimation of spatial autoregressive models. *International Regional Science Review*, 20:113–129, 1997.

[LeS00] James P. LeSage. Bayesian estimation of limited dependent variable spatial autoregressive models. *Geographical Analysis*, 32:19–35, 2000.

[LP00] James P. LeSage and R. Kelley Pace. Closed-form maximum likelihood estimates for spatial problems. Technical report, www.spatial-statistics.com, 2000.

[Ord75] J. K. Ord. Estimation methods for models of spatial interaction. *Journal of the American Statistical Association*, 70:120–126, 1975.

[Pacng] R. Kelley Pace. Simultaneous spatial and functional form transformations. In L. Anselin and J. G. M. Florax, editors, *Advances in Spatial Econometrics*. Springer-Verlag, forthcoming.

[PB97] R. Kelley Pace and R. Barry. Quick computation of spatial autoregressive estimators. *Geographical Analysis*, 29:232–246, 1997.

[PL00] R. Kelley Pace and James P. LeSage. Semiparametric maximum likelihood estimates of spatial dependence: The double bounded likelihood estimator (dble). Technical report, www.spatial-statistics.com, 2000.

[PW92] R. A. Pollack and T. J. Wales. The likelihood dominance criterion: A new approach to model selection. *Journal of Econometrics*, 47:227–242, 1992.

[PZ00] R. Kelley Pace and Dongya Zou. Closed-form maximum likelihood estimates of nearest neighbor spatial dependence. *Geographical Analysis*, 32:154–172, 2000.

Chapter 25

SPARC: SPATIAL ASSOCIATION RULE-BASED CLASSIFICATION

Jiawei Han, Anthony K. H. Tung, and Jing He

Abstract Spatial classification is to classify spatial objects based on the spatial and nonspatial features of these objects in a database. The classification results, taken as the models for the data, can be used for better understanding of the relationships among the objects in the database and for prediction of characteristics and features of new objects. Spatial classification is a challenging task due to the sparsity of spatial features which leads to high dimensionality and also the "*curse of dimensionality*". In this paper, we introduce an association-based spatial classification algorithm, called SPARC (**SP**atial **A**ssociation **R**ule-based **C**lassification), for efficient spatial classification in large geospatial databases. SPARC explores spatial association-based classification and integrates a few important techniques developed in spatial indexing and data mining to achieve high scalability when classifying a large number of spatial data objects. These techniques include micro-clustering, spatial join indexing, feature reduction by frequent pattern mining, and association-based classification. Our performance study shows that SPARC is efficient for classification of spatial objects in large databases.

Keywords: Spatial data mining, spatial classification, association-based classification, clustering, frequent pattern mining, spatial join index.

1. Introduction

With the popular use of World-Wide Web, satellite and remote sensing systems, mobile computing systems, telemetry systems, and spatial and relational database systems, huge amounts of spatial data have been collected and stored in large databases. An important emerging task is to develop powerful and efficient spatial data mining methods and tools so that the spatial data stored can be fully analyzed, explored, and utilized.

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 461–485.
© 2001 *Kluwer Academic Publishers.*

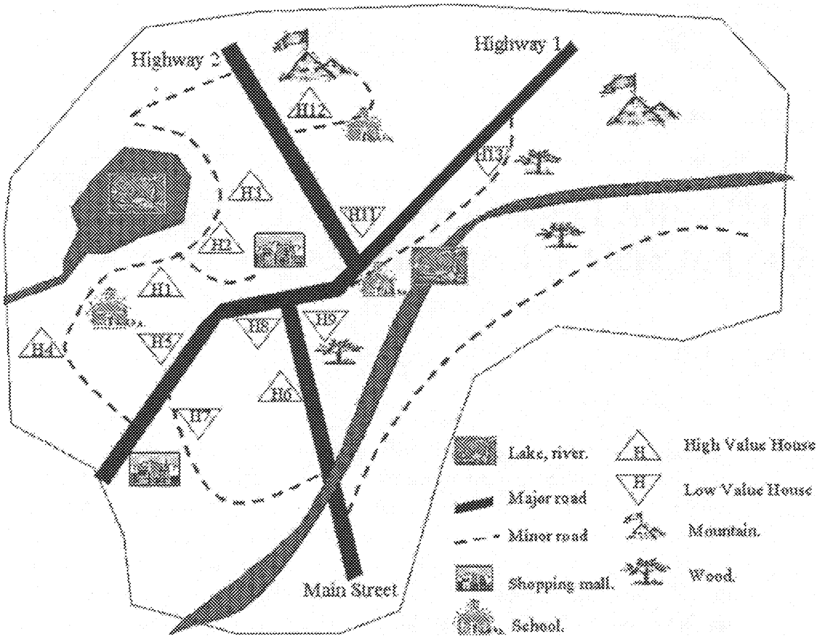

Highway 2
Highway 1
H12
H13
H3
H11
H2
H1
H4
H5
H8
H9
H6
H7
Main Street
Lake, river.
Major road
Minor road
Shopping mall.
School.
H
High Value House
H
Low Value House
Mountain.
Wood.

Figure 1. Determining the value of housing.

Among all the spatial data analysis and mining tasks, spatial classification has its special importance because it can be used to analyze the essential features of different classes of spatial objects and classify new data objects. Spatial classification is to construct and refine classification models or rules based on a set of training and test data and the models so constructed can be used for prediction.

There have been many effective classification methods developed in data mining research, such as decision trees, Bayesian networks, neural networks, etc. However, adapting these methods for classification of spatial objects is a non-trivial process because spatial features are usually sparse and high dimensional which cannot be handled effectively nor efficiently by many existing classification methods.

Let's examine one example.

Example 1.1 Figure 1 illustrates two classes of houses, high-valued vs. low-valued, surrounded by other spatial objects like lakes, roads, schools, and woods in their neighborhood. A house could be valued high if it is near a school and a park, but could be valued low if it is near a school and a busy road. Similarly, a house may be valued high if it is not far from a highway exit but its value reduced substantially, if it is too close to a busy high way due to the noise and air pollution.

To put a spatial object, such as a house, into an appropriate class, the properties of the object itself and its neighboring spatial objects should be taken into consideration. Since there could be many different objects relating to a spatial object in different ways, e.g., a house may be related to a lake, a school, or a highway, and there could be multiple spatial predicates relating different spatial objects, such as intersect, inside, connect, near-by, etc. [Ege91, KHS98], such a situation leads to high sparsity and high dimensionality in the relationships among spatial objects. This poses a great challenge to spatial classification.

Many recent studies treat spatial classification as a spatial data mining task [EKS97, EFKS98, KHS98]. For example, for effective spatial classification, [KHS98] proposes to analyze the neighborhood of each individual spatial object being classified and construct a relational table storing such information for each of such objects. A decision tree is then built using the relational table. With many spatial and nonspatial relationships being considered, the number of columns in such a table is likely to be too large to be analyzed effectively.

In this paper, we propose a new classification algorithm, called **SPARC** (**SP**atial **A**ssociation **R**ule-based **C**lassification), which performs spatial classification using spatial association rules mined in large databases [KH95, LHM98]. The major difference from the previous approaches on spatial classification is that instead of performing classification using traditional decision-tree, Bayesian, or neural network methods, SPARC develops a spatial association-based classification method. Moreover, different from mining spatial association rules by on-line progressive refinement analysis [KH95], a micro-clustering-based spatial join indexing method is applied for efficient mining of spatial associations in SPARC. In this context, SPARC integrates multiple spatial database and data mining techniques, including micro-clustering [ZRL96], spatial join indexing [Rot91, LH92], frequent itemset mining [HPY00], spatial association mining [KH95], and association-based classification [LHM98, WZH00]. Such an integration leads to an effective, high performance spatial classification process.

The organization of this paper is as follows. In Section 2, a problem description and an overall problem analysis are presented. In Section 3, a micro-clustering based spatial join indexing method is proposed. The SPARC algorithm is introduced in Section 4. A performance study on SPARC and several other spatial classification methods are reported in

Section 5. A discussion on related work and various ways to improve SPARC is in Section 6. Finally, our study is concluded in Section 7.

2. Problem Description and Analysis

We assume that the input data in our problem consists of the following:

- A set of spatial objects, $O_c = \{o_1, ..., o_n\}$, each associated with a set of spatial attributes, a set of non-spatial attributes $\{A_1, \ldots, A_m\}$, and a class label. These objects are called **training objects** since class labels are given and their associated properties are used to train the classifier which constructs a model for classification.
- Another set of spatial objects, O_r, which are called **reference objects**, since the spatial relationships between the objects in O_r and those in O_c are used to build up referential spatial relationships for spatial classification. The objects in O_r are usually of general type, such as road, school, river, lake, park, cross-section, etc. and there may exist concept hierarchies for each type, for example, a road may be classified as *major highway, local highway, street*, and *lane*, and it may have a concrete road/street name at an even lower level.
- A set of **spatial predicates** which are used to specify relationships between spatial objects. They can be classified into the following categories,
    - topological predicates, such as *disjoint, intersect, inside*, etc.
    - metric relations, such as *close_to.*
    - direction orientations, such as *south_of, west_of.*

Given the input data, the **problem of spatial classification** is to compute a classification model, which can then be used to effectively classify new spatial objects.

Analysis. The following analysis of the problem guides us in the development of spatial association-rule based classification method.

1 Association-based classification.

Since each training object is associated with a set of spatial and nonspatial features, and such features, especially the spatial ones, are of high dimensionality in nature. For example, different spatial predicates, such as *close_to*(_, *school*), *close_to*(_, *park*) and so on, are considered as different dimensions, and thus the data is of high dimensionality but is very sparse. Such kinds of data creates difficulties for traditional classification methods. Association-based classification has been shown effective at handling such data [LHM98] and thus we adopt association-based classification in our study.

2 Efficient computation of spatial associations using join index.

In association-based classification the association rules to be computed are of the form $P \to C$, where P is a conjunction of spatial and/or nonspatial predicates, and C is one of the defined classes.

Since the nonspatial predicate, such as $age(house, 15\_20\_years)$, often involves only a simple table look-up, the major challenge in the computation of spatial associations becomes how to compute spatial predicates efficiently. For example, spatial predicates may have to be evaluated repeatedly in order to derive a rule like, "*is_a(X, house)* $\wedge$ *close_to(X, school)* $\wedge$ *touch(X, park)* $\to$ $High\_Value(X)$ [1%, 80%]", which means "*if X is a house, and it is close to a school and touches a park, then it is highly valued with 1% support and 80% confidence*".

The two basic measures, *support* and *confidence*, may need some more explanation. Since the class of an object is unlikely to be determined by objects which are far away, we will only consider reference objects which are within a certain distance D from a training object when the association rules are computed. To ensure that the impact of a rule is significant in the spatial database, two measures, **support** and **confidence**, are introduced in a similar way as [AS94]. The **support** of the rule is the percentage of training objects which satisfy the rule, while the **confidence** of the rule is the probability of a training object being class C when P on the left hand side is true. For a rule to be significant, both of its support and confidence must be above two minimum thresholds, min_sup and min_conf, respectively.

For efficient computation of spatial association rules, instead of repeatedly evaluation of spatial predicates on the fly, we can precompute such spatial predicates and saved such pairs in the form of spatial join index. Later, the computation of spatial predicates in association mining will involve merely index look-up, which will save a lot of on-line, expensive spatial computation.

3 Efficient computation of information-based spatial join index by micro-clustering-based join indexing.

Since many neighboring training objects share similar spatial properties related to the same set of reference objects, instead of computing and saving each such training object, it could be more efficient in both time and space to group such training objects into small clusters (which we call *micro-clusters*) and perform micro-clustering-based spatial join index computation and reference. This leads to the development of micro-clustering-based spatial join index structure.

In the following sections, our algorithm development proceeds step-by-step in the order of computation: (1) micro-clustering-based spatial join indexing, (2) efficient computation of spatial associations using join index, and (3) spatial association-based classification.

3. Micro-Clustering-Based Spatial Join Indexing

As analyzed in Section 2, it is expensive to repeatedly compute spatial predicates and it is cost-effective to precompute spatial predicates which are likely to be repeatedly used in association mining. The precomputed results can be stored in the form of information-associated spatial join index structure [Rot91, LH92], which avoids expensive on-the-fly computation and facilitates fast retrieval.

An information-associated spatial join index record, which stores information required for association mining, could be in the form of

$$(training\_object, reference\_object, distance, set\_of\_predicates)$$

where *distance* is the distance between the $training\_object$ and the $reference\_object$, and the *set_of_predicates* is the set of spatial predicates which are evaluated true for the $training\_object$ and the $reference\_object$.

Since there still exist a large number of training objects and reference objects, there are two methods to further reduce the size of and the building cost of spatial join index.

1 Since the class of an object is unlikely to be determined by objects which are far away, only a training object and its *nearby reference_objects* will be used in the construction of spatial join index; and

2 Since a group of neighborhood training objects often share similar spatial relationships with a reference object, it is cost-effective to cluster such neighborhood training objects into one, called micro-cluster, in the study of their relationships with some reference objects.

In order to examine only the neighborhood pairs of training and reference objects and perform effective micro-clustering for efficient computation of spatial join index, a data structure called *clustering feature tree* (**CF-tree**) [ZRL96] is used, which represents each small subcluster by a triplet, called *clustering feature (CF)*. Given N d-dimensional points or objects $\{X_i\}$ in a cluster, the CF of a subcluster is defined as

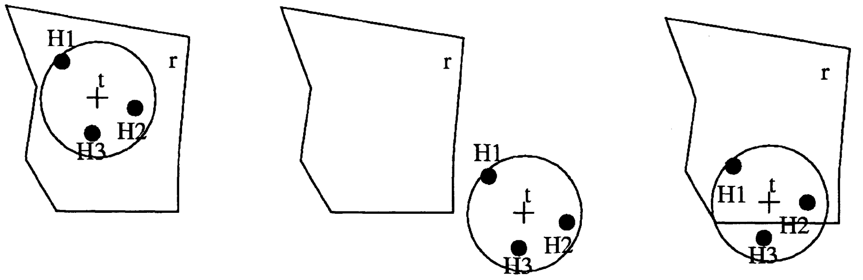
$$CF = (N, \vec{LS}, SS), \tag{3.1}$$

where N is the number of points in the subcluster, $\vec{LS}$ is the linear sum on N points, i.e., $\sum_{i=1}^{N} \vec{X_i}$, and SS is the square sum of data points, i.e., $\sum_{i=1}^{N} \vec{X_i}^2$.

A *CF-tree* is a height-balanced tree which stores the CFs for hierarchical clustering. A non-leaf node stores the sum of the CFs of their children, and thus, summarizes clustering information about their children. A CF tree has two parameters: *branching factor*, B, and *threshold*, T. The branching factor specifies the maximum number of children per non-leaf node, and the threshold parameter specifies the maximum diameter D of subclusters stored at the leaf nodes of the tree. These two parameters influence the size of the resulting tree.

In the context of classification of houses as in our example, a training object is represented as a *point* in the spatial database, and also in the CF-tree; and a reference object, since it could be a landmark, a park, a river, or a high way, etc. is presented as either a *point*, a *polyline*, or a *polygon*, in the spatial database, and also in the CF-tree. The distance between a training object o_t and a reference object o_r is the distance between the two points if the reference object o_r is a point. It is the minimum distance of o_t to o_r if o_r is a polyline. It is the minimum distance of o_t to the polygon o_r if o_t is outside the polygon o_r, or 0 if o_t is on the polygon o_r, or the negative number of the minimum distance of o_t to the polygon o_r if o_t is inside the polygon o_r.

The process of insertion of the training objects and the reference objects into CF-tree can be considered as the process of micro-clustering. A CF-tree is built dynamically as the objects are inserted. An object is inserted to the closest leaf entry (the subcluster). If the diameter of the subcluster stored in the leaf node after insertion is larger than the threshold value, then the leaf node and possibly other nodes are split. After the insertion of the new object, information about it is passed towards the root of the tree. The size of the CF tree can be changed by modifying the threshold. If a threshold value results in a CF-tree that could not fit in the main memory, the threshold is increased, and the CF-tree is rebuilt starting from the root of the tree. Thus, the process



(1) All training objects of t are inside r.

(2) All training objects of t are NOT inside r.

(3) Only some training objects of t are inside r.

Figure 2. Three cases between a training object t and a reference object r for the spatial predicate *inside*.

of rebuilding the tree is done without the necessity of re-reading all of the objects or points. As such the database is only scanned once when building the CF tree.

Having inserted the spatial objects into the CF-tree, one can compute spatial join index by evaluating the spatial predicates between each reference object and its corresponding training objects as follows.

Let the set of training objects in a leaf node l (which is treated as a *micro-cluster*) be Cl_l, and let the reference object be o_r. Evaluation is performed only on the pairs of cluster Cl_l and reference object o_r whose distance is within the distance threshold. The evaluation of each spatial predicate p has the following rules.

1 If the spatial predicate p taking Cl_l and o_r as arguments (i.e., $p(Cl_l, o_r)$) is evaluated to true, then insert this predicate as a member of the set of predicates of the join index entry $\langle Cl_l, o_r \rangle$;

2 If the spatial predicate p taking Cl_l and o_r as arguments (i.e., $p(Cl_l, o_r)$) is evaluated to false, then ignore it;

3 Otherwise, the spatial predicate p has to be evaluated at the granularity of individual objects, that is, for each object o in Cl_l, evaluate $p(o, o_r)$: If $p(o, o_r)$ is true, then insert it as a member of the set of predicates of the join index entry $\langle o, o_r \rangle$.

Example 3.1 For the evaluation of spatial predicate *inside*, Figure 2 shows three possible cases between a training object cluster t and a reference object r. Suppose t and r are close enough to be evaluated. There are three possible results for this evaluation: (1) t (e.g., a cluster of houses) is completely inside the reference object r (e.g., a city block): in this case, a predicate $inside(t, r)$ is inserted into the set of predicates of the join index entry $\langle t, r\rangle$; (2) t is completely outside the reference object r: in this case, predicate $inside(t, r)$ is false, and will not be inserted into $\langle t, r\rangle$; and (3) some objects in t are inside but some are outside r: in this case, the predicate is evaluated for each object o in t, and if it is true, $inside(o, r)$ is inserted into the join index entry $\langle o, r\rangle$.

As a summary, we have the following algorithm for computing spatial join index by micro-clustering.

Algorithm 3.1 Computation of spatial join index by micro-clustering.

Input. (1) A set of training objects O_t, (2) a set of reference objects O_r, (3) a set of spatial predicates, P (to be materialized), and (4) a distance threshold D (beyond which no predicate need to be evaluated).
Output. Spatial join index entries for O_t, O_r with the computation of P under D.
Method.

```
1) Construct CF-tree by inserting the objects in $\{O_t, O_r\}$ into the
      CF-tree $T$;
2) for each leaf $Cl_1$ in CF-tree $T$ do
3)     for each ref. object $o_r$ in leaf $Cl_2$ within distance $D$ from $Cl_1$ do
4)         for each predicate $p$ in $P$ do
5)             if predicate $p(Cl_1, o_r)$ is true
6)             then insert $p(Cl_1, o_r)$ into the predicate set of
                   the index node $(\langle Cl_1, o_r\rangle, distance(Cl_1, o_r))$;
7)             else for each object $o$ in $Cl_1$ do
8)                 if predicate $p(o, o_r)$ is true
9)                 then insert $p(o, o_r)$ into the predicate set of
                       the index node $(\langle o, o_r\rangle, distance(o, o_r))$;
```

Rationale. The algorithm computes the complete set of information-associated spatial join index for the training object set O_t and the reference object set O_r, with the set of predicate P evaluated under the distance threshold D. This is based on the following reasoning. Line (1) constructs a CF-tree by inserting the objects in $\{O_t, O_r\}$ into the CF-tree T. Based on the method for construction of the CF-tree T [ZRL96], all the objects are inserted into the CF-tree T, and the neighboring objects are inserted into the same or neighboring leaf nodes of

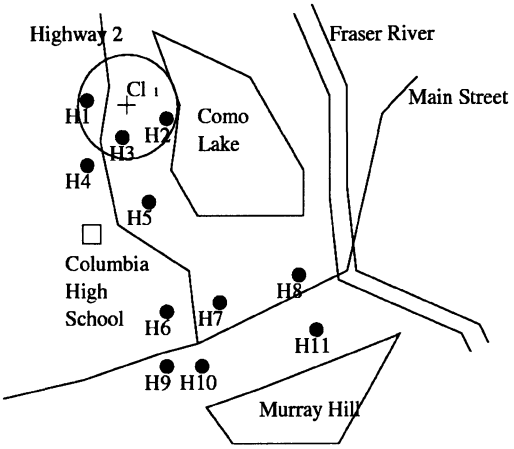


Figure 3. A set of spatial training objects (houses) and their neighborhood reference objects.

T. Lines (2)–(4) claims the procedure examines for each possible predicate p in P all the possible pairs of Cl_1 and o_r within distance D from Cl_1. This ensures all the possible predicates and object/reference pairs and the training objects are examined at the granularity of subcluster when possible, and otherwise dig deeper into the granularity of individual objects. Lines (5)–(8) evaluate such predicate at the micro-cluster granularity level when possible, and when being evaluated to true, insert it into the corresponding join index entry, but when the predicate cannot be asserted at the micro-cluster level, the predicate is evaluated for each individual training object, and when evaluated to be true, being inserted into the corresponding join index entry. Thus the algorithm computes correctly the complete set of spatial join index entries.

Let's examine how to construct spatial join index by Algorithm 3.1 using an example.

Example 3.2 Figure 3 shows a set of spatial training objects (houses) and their neighborhood reference objects: lakes, hills, highways, schools, streets, parks, and so on. The input data contains both spatial and non-spatial data. Table 1 stores the information about the training objects, and Table 2 stores that about the reference objects.

The computation of spatial join index is performed as follows.

Table 1. Information about training objects (houses)

HouseID	Spatial_loc	Years_old	Floor_size	Number_of_rooms	...	class
H1	I2345	3	3000	4	...	High
H2	A5679	1	1500	2	...	High
H3	B7563	20	2000	5	...	Low
...	...	...	...	...	...	...

Table 2. Information about reference objects

ReferenceID	Type	Name	Spatial_loc	Spatial_type	...
R9857	highway	Route No. 1	H3259	polyline	...
L5816	lake	Como Lake	L8725	polygon	...
P9715	school	Columbia High School	P6305	polygon	...
...	...	...	...	...	...

1 Construct CF-tree by inserting the objects in $\{O_t, O_r\}$ into the CF-tree T, which is performed by scanning both tables, Table 1 and Table 2, once. This is essentially micro-clustering since the set of training objects in each leaf node of the CF-tree, due to their spatial closeness, can be treated as one micro-cluster. The CF-tree T obtained by this process is shown in Figure 4.

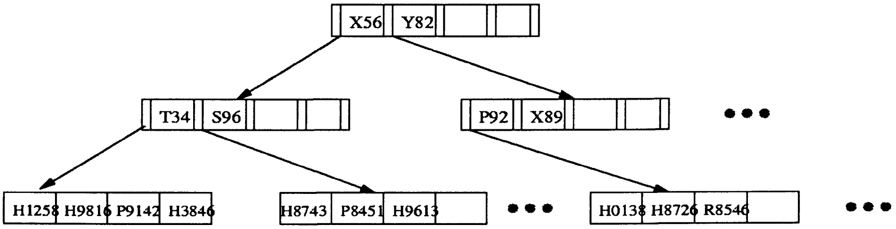


Figure 4. A fragment of the CF-tree constructed from the spatial objects.

2 Then for each leaf Cl_1 in the CF-tree T, for each reference object o_r in leaf Cl_2 (which could be the same or neighboring leaf nodes in the CF-tree (within distance D from Cl_1—which is performed by first calculating the distance between them) and the spatial predicates as follows. For each predicate p in the set of spatial predicates to be computed, such as *inside*, if *inside*(Cl_1, o_r) is true, then insert it into the predicate set of the index node $\langle Cl_1, o_r \rangle$. Otherwise,

if some of the objects, such as H_1 in Cl_1, and $inside(H_1, o_r)$ is true, then insert $p(H_1, o_r)$ into the index node $\langle H_1, o_r \rangle$. After this computation, Table 3 shows a fragment of the constructed spatial join index table, which will be used for mining spatial association and classification rules.

Table 3. Information-associated spatial join index

Train_obj	Train_obj_type	Ref_obj	Ref_obj_type	dist.	spatial_predicates
Cl_1	cluster	L9102	highway	0.35	{west_of, close_to}
Cl_2	cluster	R4592	lake	0.15	{side_of, close_to}
$H9812$	house	P4592	lake	0.05	{touch, west_of}
...	...	...	...	...	...

4. Algorithm SPARC

With the computed spatial join index, the SPARC (SPatial Association-Rule based Classification) algorithm consists of two major steps: **association rule mining** and **association-based classification.**

4.1. Association Rule Mining

After computing the spatial join index, the next step is to mine association rules that link the features of training objects with the class labels, with sufficient support and confidence. As shown in most studies of association rule mining [AS94, HPY00], instead of trying to satisfy two thresholds, *minimum support* and *minimum confidence*, at the same time, it is efficient to first mine patterns that have sufficient support and then filter out those that cannot meet the minimum confidence threshold.

To mine patterns with sufficient support, an algorithm called Apriori [AS94] can be adapted [1]. The general approach of Apriori is to mine the frequent patterns of length k ($k > 1$) from the already discovered frequent patterns of length $(k-1)$, based on the Apriori property: *if a length k pattern is frequent, all of its length $(k-1)$ patterns must be frequent.* Based on this property, candidate patterns of length k are generated from the set of frequent patterns of length $(k-1)$ that are obtained at the last iteration, and only the so generated candidates will need to be tested (i.e., with the counts accumulated and later tested to see whether it passes the minimum support threshold) in the current iteration. This candidate generation-and-test approach avoids the examination of a large number of patterns that are not candidates while preserving the completeness of the mining results.

Example 4.1 We examine the association rule mining in our running database, which is the database obtained by micro-clustering and information-associated spatial join indexing, as shown in Table 3. At the first iteration, all the frequent 1-itemsets are mined. They are used to form candidate 2-itemsets and such candidates are examined at the second iteration. Then the obtained frequent 2-itemsets are used to form candidate 3-itemsets which are examined at the third iteration, and so on. Moreover, class label is treated as a special kind of frequent pattern, and each pattern generated will have the corresponding count attached to each class in the mining. Only those frequent patterns whose frequency count associated with any class cannot pass the minimum support will not need to be examined further. All the patterns containing class labels with frequency no less than the minimum support will be used in association rule generation.

For example, at the third iteration, suppose we have the following set of three frequent patterns: {along(X, highway), close_to(X, school): (high: 199, low: 101)}, {close_to(X, school), next_to(X, park): (high: 220, low: 40)}, and {along(X, highway), next_to(X, park): (high: 150, low: 50)}, generated at the second iteration, then the pattern {along(X, highway), close_to(X, school), next_to(X, park)} is generated as a candidate pattern, which is examined with counts accumulated for each class at the fourth iteration. This may lead to the pattern: {along(X, highway), close_to(X, school), next_to(X, park), (high: 80, low: 20)}. As long as one of these class counts is above the support threshold, the combination will be considered as frequent and will be used to generate the next level of candidate combinations.

Having generated the set of frequent patterns, the computation of confidence is trivial since the count for each class is kept as our basic statistics. For example, {along(X, highway), close_to(X, school), next_to(X, park), (high: 80, low: 20)} means that the confidence for the class of high-valued house should be 80% while that for the low valued houses should be 20%. Thus we have two rules below,

along(X, highway), close_to(X, school), next_to(X, park) →
high_valued_house(X) [0.008, 0.8]

along(X, highway), close_to(X, school), next_to(X, park) →
low_valued_house(X) [0.002, 0.2]

4.2. Association-Based Classification

Having mined the rules that satisfy the minimum support and minimum confidence thresholds, the next step, association-based classification, is to develop a classification scheme which can be used to classify future class objects. A method similar to [LHM98] is adopted for this purpose.

First, the association rules are ordered according to the following definition.

Definition 4.1 (Rules Precedence) *Given two rules r_i and r_j, $r_j \prec r_i$ (also called r_i precedes r_j or r_i has higher precedence than r_j) if*

1 *the confidence of r_i is greater than that of r_j.*

2 *their confidences are the same, but the support of r_i is greater than that of r_j, or*

3 *both the confidences and supports of r_i and r_j are the same, but r_i contains less predicates than r_j on the left hand side of the rule.*

Using the order imposed by such a definition, a ranking of the rules can be generated, and each rule is associated with a unique rank among all the rules. Based on this ranking, each training object will be **covered**[2] by a rule that has the highest ranking. The most frequently occurring class is selected as the default class for training objects that are not covered by any rules. To faciitate efficient processing, an initial pruning is done by removing rules that does not cover any cases or the rules with low confidence (less than a pessimistic error-rate pruning threshold similar to [LHM98]). The remaining rules are then added one at a time into the classifier, C, until the error rate of C increase instead of decrease when a rule is added. When this happened, all remaining rules are pruned off. Using this classifier, a new spatial object can be classified by finding the highest ranked rule that covers the object. Studies in [LHM98] shows that such a classification scheme can achieve good accuracy while keeping the size of the classifier small.

A refined association rule-based model construction method has recently been proposed in [Li01], which uses the majority consensus of multiple rules to predict class labels. Such a method leads to higher classification accuracy, which has been adopted in our spatial association-rule based classification. The technical details are in [Li01] whereas our presentation here still follows the method similar to [LHM98] due to limited space.

Example 4.2 Based on the association rules mined in Example 4.1, one can sort all the rules based on their relative confidence. Since rule (4.1) has higher confidence over rule (4.1), rule (4.1) has higher ranking than rule (4.1), and thus the set of patterns matching the left hand side of the rule, "$along(X, highway), close\_to(X, school), next\_to(X, park)$" will be classified as $high\_valued\_house$.

4.3. Algorithm SPARC

As a summary of the above discussion, the Algorithm SPARC can be presented as follows.

Algorithm 4.1 Spatial association-rule based classification using micro-clustering-based join index.

Input. (1) An information-based join index structure JI, and (2) input table(s) which store information related to training objects.

Output. A spatial association-based classification model.

Method.

The classification is performed in two steps: (1) mining class-information attached spatial association rules, and (2) constructing classification models based on such association rules.

```
// First, mine class-information attached spatial association rules
$L_1$ = the set of frequent spatial/nonspatial single predicate
        derived from JI and other input information table;
for ($i = 1; L_{i-1} \neq \phi; i{+}{+}$) {
    $C_i$ = candidate_set_generation($L_{i-1}$);
    if $C_i = \emptyset$, then return;
    scan JI and other necessary tables to get count (attached to
    each class label) for each predicate set in $C_i$;
    $L_i$ = {predicate set in $C_i$ whose count attached to at least
    one class label is no less than min_supp}
}
for each predicate set $P$ containing a class $c_i$ do
    if confidence = $count(c_i)$/all the counts for $P \geq$ min_conf
    then generate rule "$P \rightarrow c_i$ [support, confidence]"

// Second, mine classification model based on mined association rules
sort the association rules derived above according to the
Rule Precedence rules specified in Definition 4.1;
Remove the association rules that have lower precedence and do
```

not cover any additional predicate sets;
12) Classification model = the set of association rules so derived;

Rationale. The algorithm derives a classification model with high accuracy, based on the following reasoning. In the first part, it mines class-information attached spatial association rules, as described from Lines (2)—(9), and whose correctness can be reasoned in a similar way as [AS94]. In the second part, it mines classification model based on the mined association rules, as described from Lines (10)—(12), and whose correctness can be reasoned in a similar way as [LHM98]. The two examples, Example 4.1 and Example 4.2, have demonstrated the association-based classification process.

5. Performance

In this section, we report our experiments and performance study on the efficiency of the SPARC algorithm in comparison with some alternative methods.

Since traditional decision tree-based or Bayesian classification algorithms cannot handle high-dimensional, high sparsity data (and this point will be further discussed in Section 6), our performance study is confined to the spatial association-based classification approach. Under this framework, we consider the following three methods: (1) simple join-index method, (2) micro-clustering based join-index method (which is SPARC), and (3) spatial association mining by progressive refinement [KH95] which mines frequent spatial predicates by performing first rough spatial computation, such as computation of frequent predicates using MBR (minimum bounding rectangle) instead of using detailed polygons, and then refined computation of polygons only for those pairs of spatial objects whose corresponding spatial predicate, such as *close_to*, is frequent. Each of the first two methods may have two variations: *precomputation of spatial join index* vs. *on-line computation of spatial join index*; whereas the third one, spatial association mining by progressive refinement, cannot perform precomputation before knowing the concrete min_support threshold which is determined by a particular mining query. Thus, we have in total the following five alternative algorithms in our performance study.

1 **JI** (Join Index): This method first performs online computation of spatial join index, and then association mining by FP-Growth [HPY00], followed by association-based classification.

2 **PR** (Progressive Refinement): This method performs coarse computation to derive predicates that are potentially frequent and

computes frequent predicates by refined spatial computation on those derived predicates, and then mines associations by FP-Growth, followed by association-based classification.

3 **MJI** (Micro-clustering based Join Index): The difference of MJI from JI is at that MJI performs micro-clustering when building join index. Micro-clustering is performed by construction of CF-trees as described in [ZRL96], with a reference to a clustering-based join index construction algorithm, Spot [Yeh99]. No precomputation of spatial join index is conducted in this case, as well as all the previous cases.

4 **JI-P** (Join Indexing with Pre-computation): This is the same as JI except that the join index entries are pre-computed. Thus the cost for computing the join index is excluded in the evaluation of its performance.

5 **MJI-P** (Micro-clustering based Join Indexing with Pre-computation): This is the same as MJI except that the join index are pre-computed. Also, the cost for computing join index entries is excluded in the evaluation of its performance.

Synthetic data are used in our experiments. The spatial references are created to represent the major roads, parks, shopping centers, beaches, hospitals and some public schools in Vancouver. Houses are randomly generated so that they are either members of clusters or outliers. We conduct a number of experiments, each set with a few parameters: the average number of members in a cluster is 10, the percentage of outliers is 10%, the number of spatial reference objects is 60, the number of houses (training objects) varies from 1,000 to 100,000.

All the experiments are conducted on a computer with Pentium-500 CPU and 128 MB main memory, running on Microsoft NT. Our experimental results are reported as follows.

Figure 5 compares the performance of the five algorithms (JI, PR, MJI, JI-P, and MJI-P) with a changing number of spatial objects. Figure 5 a) shows the running time of the five algorithms. Since MJI computes spatial predicates for less number of object pairs by micro-clustering, MJI outperforms JI and PR significantly. Since PR performs refined computation only on frequent predicates, PR is more efficient than JI. Both MJI-P and JI-P outperforms others since the expensive spatial computation are precomputed which saves the cost of on-line computation. Figure 5 b) shows the scalability test with the number of training objects grows up to 100000, where the Y-axis is on logarithmic scale in order to view the relative performance clearly. One can see that MJI scales up more gracefully than PR and JI.

Figure 6 shows the relative performance of MJI and PR, with different support thresholds when the number of spatial objects is 10000. The indicated support threshold is an absolute number. One can see that when the support threshold drops, the runtime cost of PR goes up quicker than that of MJI due to the benefit of progressive refinement in PR reduces.

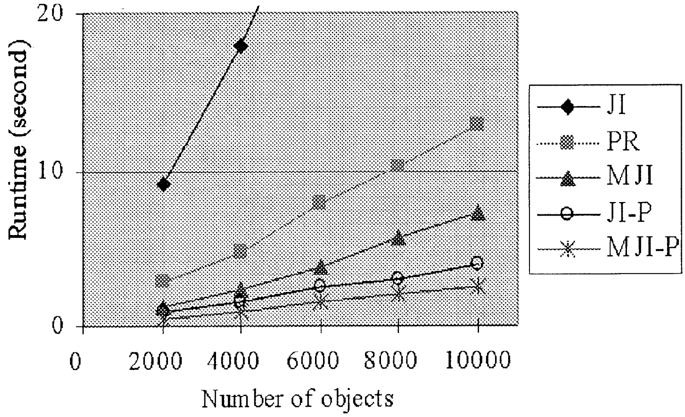
Our performance analysis shows that MJI-P has the best performance, followed by JI-P, MJI, PR, and then JI in most cases. This is analyzed as follows.

1 The major cost of spatial association mining is the evaluation of a large number of spatial predicates, that is, the computation of spatial join indicies in the case of JI's. Since both JI-P and MJI-P compute join index offline, and the pre-computed JIs are used at mining, they outperform other methods easily. Between the two, MJI is more efficient since multiple training objects in the same cluster share information about the truth/falsity of certain spatial predicates by micro-clustering.

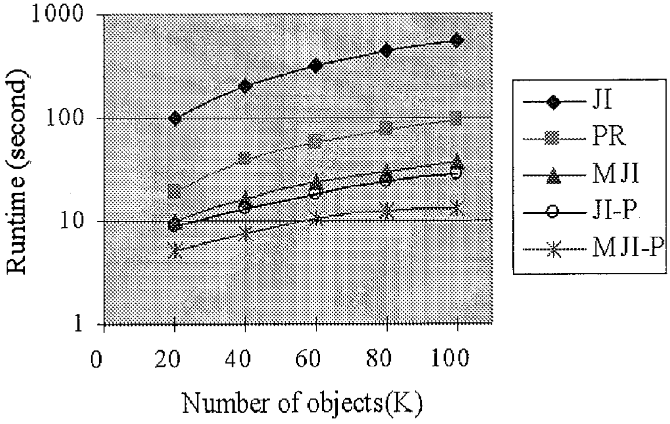
2 MJI and MJI-P outperform JI and JI-P. This is because MJI and MJI-P use micro-clusters to group spatial objects (houses), which results in savings in the predicate computation. In addition, houses within the same clusters share the same predicates, which leads to more compressed FP-tree than that of JI, thus less time for MJI and MJI-P to do association rule mining than JI and JI-P.

3 JI is outperformed by PR because PR performs fine granularity spatial computation only on those candidates found by coarse computation, while JI performs fine computation on all training-reference object pairs. Furthermore, since PR has already filtered out some infrequent predicates when it conducts association rule mining, it results in a smaller FP-tree than JI.

4 MJI, even without pre-computation, outperforms PR. This shows the effectiveness of our approach. Even though MJI may produce more predicates than PR (due to not perform filtering with the minimum support threshold), the cost is offset by its sharing of expensive predicate computation by micro-clustering.



a) Number of training objects up to 10000



b) Number of training objects up to 100000

Figure 5. Processing efficiency of the five algorithms related to the number of training objects.

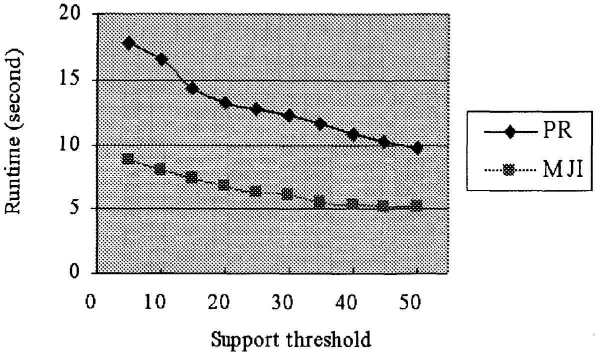


Figure 6. Processing efficiency of MJI and PR with different support thresholds.

6. Discussion

In this section, we discuss how to further improve the performance and/or quality of spatial association-based classification and also compare our approach with a few related methods.

6.1. Further improvements of efficiency and classification quality

Based on the algorithm design and performance study in the previous sections, we believe that there are several methods that may further improve the performance and/or quality of spatial association-based classification.

1 R-tree- or quadtree-based instead of CF-tree-based micro-clustering.

In our examples, the CF-tree structure [ZRL96] is used for spatial micro-clustering. CF-tree is good for spatial data consisting of multi-dimensional points. However, many spatial objects may consist of other spatial types, such as polygons, polylines, raster data, etc. The R-tree structure [Gut84, BKSS90] has been used popularly for storing vector data, such as polygons, polylines, and points; whereas the quadtree structure [Sam90] has been popularly used for storing raster data. There have been reports that the R+-tree structure is effective and efficient for spatial data mining, such as density-based clustering [EKSX96]. Based on our analysis, the R-tree or R+-tree structures as well as the quadtree structure can be used for micro-clustering in the context of association-based classification, where the leaf nodes (or those close to the leaves) can be considered as micro-clusters. The choice of which data structure to adopt should be based on the types of data to be handled in association mining.

2 Micro-clustering of reference objects as well as training objects.

In our algorithm, micro-clustering is performed on training objects only. A spatial predicate on one reference object and a group of training objects will need to be evaluated only once if these training objects are grouped into one micro-cluster. This method improves the processing efficiency. However, micro-clustering should not be confined only to training objects. Reference objects, if physically located close to each other, can be clustered as one micro-cluster in the analysis as well. Moreover, with minor modification to SPARC, a micro-cluster may contain mixed types of objects: both training and reference objects.

3 Attachment of more information to the join index structure.

Table 4. Association of more information in spatial join index.

Train_obj	class distrib.	Ref_obj	Ref_type	Ref_ subtype	dist.	spatial predicates
Cl_1	30:50	L9102	highway	local	0.35	{west_of, close_to}
Cl_2	62:10	R4592	lake	small	0.15	{side_of, close_to}
$H9812$	1:0	P4592	lake	small	0.05	{touch, west_of}
...	...	...	...	...	...	...

Additional information can be attached to information-associated join index structure to enhance the power and efficiency of association mining. For example, if the class distribution information is associated with the join index structure as shown in Table 4, there is no need to consult the training object table, Table 1, when mining associations on the spatial predicates only since the distribution of training objects in different classes are captured in the join index structure.

4 Mining associations with multi-level reference and/or training objects.

If the training object table, the reference object table, and/or the join index structure contain the multiple levels information associated with training and reference objects, such as the concrete name of the reference object (e.g., Como Lake), the category and subcategory of the reference object (e.g., lake, small-lake), associations can be mined at multiple levels of granularity, which facilitates multi-level spatial association and classification.

5 Mining associations with multi-level spatial predicates.

In SPARC, different spatial predicates are evaluated individually with no joint efforts in evaluation. Based on the nature of spatial predicates, one can promote multi-level spatial predicates, from coarse to fine, to improve the processing efficiency. For example, one can define a spatial predicate, *generalized_close_to*, which covers a set of fine predicates, such as *close_to, near, touch, intersect, contain*, and only for those micro-clusters which are *generalized_close_to* a reference object, their corresponding fine spatial predicates will need to be evaluated. Such an integration of micro-clustering with progressive refinement [KH95] may lead to further improved performance.

6.2. Related work

The SPARC classification method integrates a few data mining techniques and spatial data structures. There are some related studies on spatial association mining and association-based classification. We will compare SPARC with these previous works.

First, spatial classification analysis has been studied using some classical classification methods, such as decision trees [EKS97], Bayesian classification, and some statistics-based approaches [EKS97]. However, classification involving spatial predicates is high dimensional in nature because there are many potential spatial predicates, such as overlap, intersect, west_of, etc., and there are also many possible spatial reference objects useful for spatial classification. This leads to highly sparse, high dimensional classification space. For such space, association-based classification, such as [LHM98], has been proven effective. However, to the best of our knowledge, there has been no study on association-based spatial classification. Therefore, this study makes the first attempt in this direction.

Second, progressive refinement (PR) method has been proposed for efficient spatial association [KH95]. As shown in our performance study, micro-clustering-based spatial association proposed in this study achieves better performance than PR due to two reasons: (1) precomputation of join index provides better on-line mining efficiency due to query-independent, off-line computation of spatial relationships and storing them in compressed join index form, and (2) micro-clustering avoids computation of spatial predicates for every training-referencing pair of spatial objects by grouping processing. It is possible to integrate the two methods, MJI and PR, for improved on-line computation, as discussed in Section 6.

Finally, our method is an integration of multiple methods in classification, association, clustering, join indexing, and spatial data structures. All the previous studies on these topics contribute to the success of this technique. However, our work different from theirs is at its integration of these methods to achieve high quality and high performance spatial classification in large databases.

7. Conclusions

In this paper, we have introduced an association-based spatial classification method, SPARC, for efficient spatial association rule-based classification in large geospatial databases. Due to the high dimensionality and sparsity of spatial features associated with different spatial objects, efficient spatial classification is a challenging task. The SPARC algorithm explores spatial association-based classification and integrates a few important techniques developed in spatial indexing and data mining, which include micro-clustering, spatial join indexing, feature reduction by frequent pattern mining, and association-based classification. Our performance study shows that SPARC is efficient for classification of spatial objects in large databases.

There are many interesting issues which need to be studied further, including the use of traditional spatial data structures, such as R-trees and quad-trees for micro-clustering, integration of micro-clustering and progressive refinement in spatial association, and multi-level spatial association and classification in large databases. Also, this study also shows that it is highly desirable to construct a spatial data mining system which integrates multiple spatial data mining functions, including clustering, association, and classification for successful multi-dimensional spatial analysis.

Acknowledgments

The work was supported in part by the Natural Sciences and Engineering Research Council of Canada (grant NSERC-A3723) and the Networks of Centres of Excellence of Canada (grants GEOIDE and NCE/IRIS-3).

Notes

1. For easy understanding, we introduce only the Apriori algorithm. In our implementation, instead of using the classical Apriori algorithm, we use a more efficient mining algorithm, FP-growth [HPY00], although the remaining discussions are still valid.

2. A training object is covered by a rule r if it satisfies all the predicates on the left hand side of r and belongs to the class stated by the right hand side of r.

References

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago, Chile, Sept. 1994.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. 1990 ACM SIGMOD Int. Conf. Management of Data (SIGMOD'90)*, pages 322–331, Atlantic City, NJ, June 1990.

[EFKS98] M. Ester, A. Frommelt, H.-P. Kriegel, and J. Sander. Algorithms for claracterization and trend detection in spatial databases. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 44–50, New York, NY, Aug. 1998.

[Ege91] M.J. Egenhofer. Reasoning about binary topological relations. In *Proc. 1991 Int. Symp. Large Spatial Databases (SSD'91)*, pages 143–160, Zurich, Switzerland, Aug. 1991.

[EKS97] M. Ester, H.-P. Kriegel, and J. Sander. Spatial data mining: A database approach. In *Proc. 1997 Int. Symp. Large Spatial Databases (SSD'97)*, pages 47–66, Berlin, Germany, July 1997.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Portland, Oregon, Aug. 1996.

[Gut84] A. Guttman. R-tree: A dynamic index structure for spatial searching. In *Proc. 1984 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'84)*, pages 47–57, Boston, MA, June 1984.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pages 1–12, Dallas, TX, May 2000.

[KH95] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proc. 1995 Int. Symp. Large Spatial Databases (SSD'95)*, pages 47–66, Portland, Maine, Aug. 1995.

[KHS98] K. Koperski, J. Han, and N. Stefanovic. An efficient two-step method for classification of spatial data. In *Proc. 8th Symp. Spatial Data Handling*, pages 45–55, Vancouver, Canada, 1998.

[LH92] W. Lu and J. Han. Distance-associated join indices for spatial range search. In *Proc. 1992 Int. Conf. Data Engineering (ICDE'92)*, pages 284–292, Phoenix, AZ, Feb. 1992.

[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 80–86, New York, NY, Aug. 1998.

[Li01] W. Li. Classification based on multiple association rules. In *M.Sc. Thesis*, School of Comptuing Science, Simon Fraser University, April 2001.

[Rot91] D. Rotem. Spatial join indices. In *Proc. 1991 Int. Conf. Data Engineering (ICDE'91)*, pages 500–509, Kobe, Japan, April 1991.

[Sam90] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[WZH00] K. Wang, S. Zhou, and Y. He. Growing decision tree on support-less association rules. In *Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00)*, pages 265–269, Boston, MA, Aug. 2000.

[Yeh99] T. S. Yeh. Spot: Distance based join indices for spatial data. In *Proc. 7th Int. Symp. Advances in Geographic Information Systems*, pages 103–109, Kansas City, MO, Nov. 1999.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 103–114, Montreal, Canada, June 1996.

Chapter 26

WHAT'S SPATIAL ABOUT SPATIAL DATA MINING: THREE CASE STUDIES

Shashi Shekhar, Yan Huang, Weili Wu, C.T. Lu, and S. Chawla

Abstract Spatial data mining is the process of discovering interesting and previously unknown, but potentially useful, patterns from large spatial datasets. Extracting interesting and useful patterns from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation. A popular approach is to apply classical data mining techniques after transforming spatial components into non-spatial components via feature selection. An alternative is to explore new models, new objective functions, and new patterns which are more suitable for spatial data and their unique properties. This chapter investigates techniques in the literature to incorporate spatial components via feature selection, new models, new objective functions, and new patterns.

Keywords: spatial data mining, feature selection, spatial databases, co-location rules, spatial autocorrelation, spatial outliers

1. Introduction

Widespread use of spatial databases [Gut94, SC01, SCR+99, Wor95] is leading to an increasing interest in mining interesting and useful, but implicit, spatial patterns [Gre00, KAH96, Mar99, RS99, SNM+95]. Spatial data sets and patterns are abundant in many application domains related to NASA, the National Imagery and Mapping Agency(NIMA), the National Cancer Institute(NCI), and the United States Department of Transportation(USDOT). Efficient tools for extracting information from geo-spatial data are crucial to organizations which make decisions based on large spatial data sets. These organizations are spread across many domains including ecology and environment management, pub-

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 487–514.
© 2001 *Kluwer Academic Publishers.*

lic safety, transportation, public health, business, travel, and tourism [AM95, HGL93, IES89, Kru95, Hai89, SYH93, SNM$^+$95, YL97].

Extracting interesting and useful patterns from spatial datasets is more difficult than extracting corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation.

There are different representations for spatial data. Because many datasets are collected from satellites, aerial-photographs, and Digital Elevation Maps (DEM), much spatial information is stored as thematic maps which are qualitative and categorical **raster** data, usually maps of land cover classes such as temperature, rainfall, pasture, urban areas, and standing water. Coupled with this **raster** data representation is **vector** data, the other common GIS data model, made up of points, lines, or polygons, with associated attributes. Spatial data mining deals with not only data types such as integers, dates and strings, but also complex data types like points, lines, and polygons. Furthermore, relations between spatial objects add another dimension to the complexity of spatial data mining. Basic spatial relations include metric (e.g. distance), directional (e.g. north of), and topological (e.g. adjacent).

Traditional data mining algorithms [Agr94] often make assumptions (e.g. independent, identical distributions) which violate Tobler's first law of Geography: everything is related to everything else, but nearby things are more related than distant things [Tob79]. In other words, the values of attributes of nearby spatial objects tend to systematically affect each other. In spatial statistics, an area within statistics devoted to the analysis of spatial data, is called spatial autocorrelation [Cre93]. Scientists and researchers in several disciplines have created, adapted, and applied statistical techniques to spatial data. For example, in image processing and vision, Markov Random Fields(MRFs) is a popular model to incorporate context for image segmentation and classification. Economists use spatial autoregression models to predict and estimate trends in regional economies. The variogram, a tool to capture spatial information in data, is widely used in geography and remote sensing. In spatial data mining, knowledge discovery techniques which ignore spatial autocorrelation typically perform poorly in the presence of spatial data. The models derived may turn out to be not only biased and inconsistent, but may also be a poor fit to the data set.

In this paper, we provide three case studies in spatial data mining. The first case study examines the classification of spatial datasets. The second case study describes the generalization of association rules to spatial co-location patterns. The third case study focuses on detecting spatial outliers. In each case study, we bring out unique challenges of spatial data mining relative to classical data mining.

2. Spatial Classification

Given a set of data (a training set), the classification task is to build a model to predict one categorical attribute known as the dependent attribute based on the other attributes as accurately as possible. A set of data (different from the training set) with dependent attributes known in advance is used to validate the model. In spatial classification, the membership of an object may be affected by the memberships of its neighboring objects.

2.1. An Illustrative Application Domain

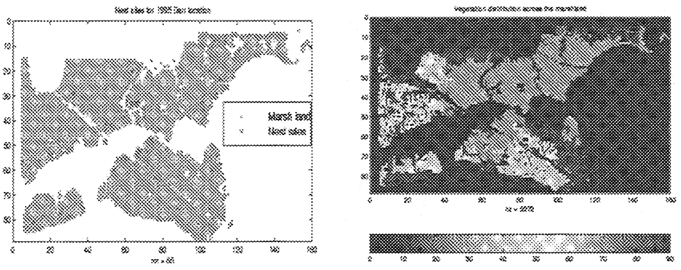
We now introduce an example to illustrate the different concepts in spatial data mining. We are given data about two wetlands, named Darr and Stubble, on the shores of Lake Erie in Ohio USA in order to *predict* the spatial distribution of a marsh-breeding bird, the red-winged blackbird (*Agelaius phoeniceus*). The data was collected from April to June in two successive years, 1995 and 1996.

A uniform grid was imposed on the two wetlands and different types of measurements were recorded at each cell or pixel. In total, values of seven attributes were recorded at each cell. Domain knowledge is crucial in deciding which attributes are important and which are not. For example, *Vegetation Durability* was chosen over *Vegetation Species* because specialized knowledge about the bird-nesting habits of the red-winged blackbird suggested that the choice of nest location is more dependent on plant structure, plant resistance to wind, and wave action than on the plant species.

Our goal is to build a model for predicting the location of bird nests in the wetlands. Typically the model is built using a portion of the data, called the **Learning** or **Training** data, and then tested on the remainder of the data, called the **Testing** data. In the learning data, all the attributes are used to build the model and in the testing data, one value is *hidden*, in our case the location of the nests.

We focus on three independent attributes, namely *Vegetation Durability*, *Distance to Open Water*, and *Water Depth*. The spatial distribution of *Vegetation Durability* and the actual nest locations for the Darr wetland in 1995 are shown in Figure 1. These maps illustrate the following two important properties inherent in spatial data.

1 The values of *Vegetation Durability* and the actual nest locations which are referenced by spatial location tend to vary gradually



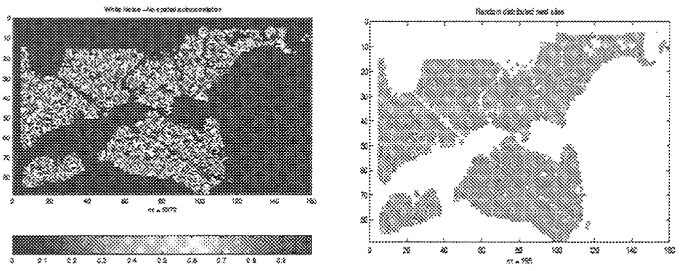
(a) Nest Locations (b) Vegetation Durability

Figure 1. (a) Learning dataset: The geometry of the wetland and the locations of the nests, (b) The spatial distribution of *vegetation durability* over the marshland

over space (values of *Distance to Open Water* and *Water Depth* are similar). While this may seem obvious, classical data mining techniques, either explicitly or implicitly, assume that the data is *independently* generated. For example, the maps in Figure 2 show the spatial distribution of attributes if they were independently generated. Classical data mining techniques like logistic regression [OM97] and neural networks [OO99] were applied to build spatial habitat models. Logistic regression was used because the dependent variable is binary (nest/no-nest) and the logistic function "squashes" the real line onto the unit-interval. The values in the unit-interval can then be interpreted as probabilities. The study concluded that with the use of logistic regression, the nests could be classified at a rate 24% better than random [OO99].

2 The spatial distributions of attributes sometimes have distinct local trends which contradict the global trends. This is seen most vividly in Figure 1(b), where the spatial distribution of *Vegetation Durability* is jagged in the western section of the wetland as compared to the overall impression of uniformity across the wetland. This property is called spatial heterogeneity.

The fact that classical data mining techniques ignore spatial autocorrelation and spatial heterogeneity in the model-building process is one reason why these techniques do a poor job. A second, more subtle but equally important reason is related to the choice of the objective function to measure classification accuracy. For a two-class problem, the standard way to measure classification accuracy is to calculate the percentage of correctly classified objects. However, this measure may



(a) pixel property with independent identical distribution

(b) Random nest locations

Figure 2. Spatial distribution satisfying random distribution assumptions of classical regression

not be the most suitable in a spatial context. *Spatial accuracy*—how far the predictions are from the actuals—is as important in this application domain due to the effects of the discretizations of a continuous wetland into discrete pixels, as shown in Figure 3. Figure 3(a) shows the actual locations of nests and 3(b) shows the pixels with actual nests. Note the loss of information during the discretization of continuous space into pixels. Many nest locations barely fall within the pixels labeled 'A' and are quite close to other blank pixels, which represent 'no-nest'. Now consider two predictions shown in Figure 3(c) and 3(d). Domain scientists prefer prediction 3(d) over 3(c), since the predicted nest locations are closer on average to some actual nest locations. The classification accuracy measure cannot distinguish between 3(c) and 3(d), and a measure of spatial accuracy is needed to capture this preference.

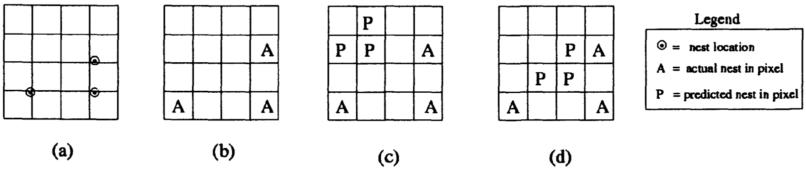


Figure 3. (a)The actual locations of nests, (b)Pixels with actual nests, (c)Location predicted by a model, (d)Location predicted by another model. Prediction(d) is spatially more accurate than (c).

A simple and intuitive measure of spatial accuracy is the Average Distance to Nearest Prediction (ADNP) from the actual nest sites, which can be defined as

$$ADNP(A, P) = \frac{1}{K}\sum_{k=1}^{K} d(A_k, A_k.nearest(P)).$$

Here A_k represents the actual nest locations, P is the map layer of predicted nest locations and $A_k.nearest(P)$ denotes the nearest predicted location to A_k. K is the number of actual nest sites.

2.2. Approaches of Modeling Spatial Dependencies

Decision Tree Approaches. A spatial classification algorithm is proposed in [EKS97]. This method is based on the well-known ID3 [Qui86] algorithm. It employs the "neighborhood" concept and extends the attributes by considering not only properties of the classified objects, but also the attribute values of neighboring objects. Objects are considered neighbors if they satisfy some neighborhood relations such as *overlap, close-to, east, etc.* For instance the economic power of the city can be classified based on the types of neighboring cities. In this example, the dependent attribute is *economic power.* Both the attributes of object city (*e.g. population of city, amount of taxes of city*) and the attributes of neighborhing cities (*e.g. type of neighbor of city, type of neighbor of neighbor of city*) are considered by this spatial classification algorithm. The algorithm "materializes" spatial relationship as attributes, but it does not analyze spatial autocorrelation, which is an important property of the spatial data.

Kopersiki, Han, and Stefanovic presented an efficient method for spatial classification [KHS98]. The authors proposed some optimization techniques like the "Two-Step" spatial computation approach to build decision trees. This algorithm includes the spatial relationship as a predicate in decision tree. More accurate and efficient decision trees can be produced via this two-step approach where coarse computation are performed to get a sample of approximate spatial predicates followed by fine computations done only for promising patterns. However, this algorithm does not take into account spatial autocorrelation and does not explicitly factor spatial properties into the model.

Logistic Regression Modeling. Given an $n-$vector $\mathbf{y}$ of observations and an $n \times m$ matrix $\underline{X}$ of explanatory data, classical linear regression models the relationship between y and $\underline{X}$ as

$$\mathbf{y} = \mathbf{X}\beta + \epsilon.$$

Here $\mathbf{X} = [1, \underline{X}]$ and $\beta = (\beta_0, \ldots, \beta_m)^t$. The standard assumption on the error vector ϵ is that each component is generated from an independent, identical and normal distribution, i.e, $\epsilon_i = N(0, \sigma^2)$.

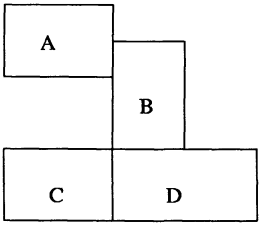
When the dependent variable is binary, as is the case in the "bird-nest" example, the model is transformed via the logistic function and the dependent variable is interpreted as the probability of finding a nest at a given location. Thus, $Prob(y = 1) = \frac{e^{X\beta}}{1+e^{X\beta}}$. This transformed model is referred to as **logistic** regression.

The fundamental limitation of classical regression modeling is that it assumes that the sample observations are independently generated. This may not be true in the case of spatial data. As we have shown in our example application, the explanatory and the independent variables show a moderate to high degree of spatial autocorrelation(see Figure 1). The inappropriateness of the independence assumption shows up in the residual errors, the ϵ_i's. When the samples are spatially related, the residual errors reveal a systematic variation over space, i.e., they exhibit high spatial autocorrelation. This is a clear indication that the model was unable to capture the spatial relationships existing in the data. Thus the model is a poor fit to the data. Incidently, the notion of spatial autocorrelation is similar to that of time autocorrelation in time series analysis but is more difficult to model because of the multi-dimensional nature of space.

Spatial Autocorrelation and Examples. Many measures are available for quantifying spatial autocorrelation. Each has strengths and weaknesses. Here we briefly describe Moran's I measure.

In most cases, Moran's I measure (henceforth MI) ranges between -1 and +1 and thus is similar to the classical measure of correlation. Intuitively, a higher absolute value indicates high spatial autocorrelation. A high positive value implies that like classes tend to cluster together or attract each other. A low negative value indicates that high and low values are interspersed. Thus like classes are de-clustered and tend to repel each other. A value close to zero is an indication that no spatial trend (random distribution) is discernible using the given measure.

All spatial autocorrelation measures are crucially dependent on the choice and design of the contiguity matrix W. The design of the matrix itself reflects the influence of neighborhood. Two common choices are the four and the eight-neighborhood. Thus given a lattice structure and a point S in the lattice, a four-neighborhood assumes that S influences all cells which share an edge with S. In an eight-neighborhood, it is assumed that S influences all cells which either share an edge or a vertex. An eight neighborhood contiguity matrix is shown in Figure 4. The contiguity



(a) Map

$$\begin{array}{c|cccc|} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 0 \\ B & 1 & 0 & 1 & 1 \\ C & 0 & 1 & 0 & 1 \\ D & 0 & 1 & 1 & 0 \\ \hline \end{array}$$

b) Contiguity Matrix W

Figure 4. A spatial neighborhood and its contiguity matrix

matrix of the uneven lattice (left) is shown on the right hand-side. The contiguity matrix plays a pivotal role in the spatial extension of the regression model.

Spatial Autoregression Model(SAR). We now show how spatial dependencies are modeled in the framework of regression analysis.

This framework may serve as a template for modeling spatial dependencies in other data mining techniques. In spatial regression, the spatial dependencies of the error term, or, the dependent variable, are directly modeled in the regression equation [Ans88]. Assume that the dependent values y'_i are related to each other, i.e., $y_i = f(y_j)\ i \neq j$. Then the regression equation can be modified as

$$\mathbf{y} = \rho W\mathbf{y} + \mathbf{X}\beta + \epsilon.$$

Here W is the neighborhood relationship contiguity matrix and ρ is a parameter that reflects the strength of spatial dependencies between the elements of the dependent variable. After the correction term $\rho W\mathbf{y}$ is introduced, the components of the residual error vector ϵ are then assumed to be generated from independent and identical standard normal distributions.

We refer to this equation as the ***Spatial Autoregressive Model (SAR)***. Notice that when $\rho = 0$, this equation collapses to the classical regression model. The benefits of modeling spatial autocorrelation are many: (1) The residual error will have much lower spatial autocorrelation, i.e., systematic variation. With the proper choice of W, the residual error should, at least theoretically, have no systematic variation. (2) If the spatial autocorrelation coefficient is statistically significant, then SAR will quantify the presence of spatial autocorrelation. It will indicate the extent to which variations in the dependent variable (y) are explained by the average of neighboring observation values. (3) Finally, the model will have a better fit, i.e., a higher R-squared statistic.

As in the case of classical regression, the SAR equation has to be transformed via the logistic function for binary dependent variables. The estimates of ρ and β can be derived using maximum likelihood theory or Bayesian statistics. We have carried out preliminary experiments using the spatial econometrics MATLAB package for making the MATLAB toolbox, which implements a Bayesian approach using sampling-based Markov Chain Monte Carlo (MCMC) methods [LeS97]. The general approach of MCMC methods is that when the joint-probability distribution is too complicated to be computed analytically, then a sufficiently large number of samples from the conditional probability distributions can be used to estimate the *statistics* of the full joint probability distribution. While this approach is very flexible and the workhorse of Bayesian statistics, it is a computationally expensive process with slow convergence properties. Furthermore, at least for non-statisticians, it is a non-trivial task to decide what "priors" to choose and what analytic expressions to use for the conditional probability distributions.

Computing Markov Random Fields with Graph Partitioning Technique. Markov Random Fields(MRFs) generalize Markov Chains to multi-dimensional structures. Since there is no natural order in a multi-dimensional space, the notion of a transition probability matrix is absent in MRFs.

MRFs have found applications in image processing and spatial statistics, where they have been used to estimate spatially varying quantities like intensity and texture for noisy measurements. Typical images are characterized by piece-wise smooth quantities, i.e, they vary smoothly but have sharp jumps(discontinuities) at the boundaries of the homogeneous areas. Because of these discontinuities, the least-squares approach does not provide an adequate framework for the estimation of these quantities. MRFs provide a mathematical framework to model our *priori* belief that spatial quantities consist of smooth patches with occasional jumps.

We will follow the approach suggested in [BVZ99] where it is shown that the maximum a posteriori estimate of a particular configuration of an MRF can be obtained by solving a suitable min-cut multi-way graph partitioning problem.

Example 1: A classification problem with no spatial constraints

Even though MRFs are inherently multi-dimensional, we will use a simple one-dimensional example to illustrate the main points. Consider the graph $G = (V, E)$ shown in Figure 5(a). The node-set V itself consists of two disjoint sets X and L. The members of X are $\{x_1, x_2, x_3\}$ and the

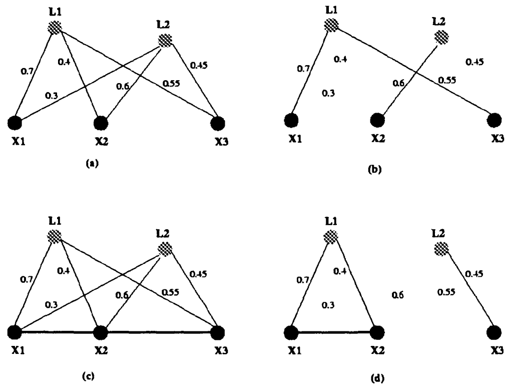


Figure 5. (a) Initially each pixel is assigned to both labels with different edge weights. The edge weights correspond to probabilities of assigning each pixel to a different label, (b) A min-cut graph partitioning induces a labeling of the pixel set. Labels which correspond to the maximum probabilities are retained, (c) `Spatial autocorrelation` is modeled by introducing edges between pixel nodes, (d) A min-cut graph partitioning does not necessarily induce a labeling where the labeling with maximum probabilities are retained. If two neighboring pixels are assigned different labels, then the edge connecting the pixels is added to the cut-set.

members of L are $\{l_1, l_2\}$. Typically the $x_i's$ are the pixels and the $l_j's$ are the labels, like *nest* or *no-nest.* There is an edge between each member of the set X and L. Here we will interpret the edge weights as probabilities. For example, $p_1 = Prob(x_1 = l_1) = 0.7$ and $p_2 = Prob(x_1 = l_2) = 0.3$; $p_1 + p_2 = 1$.

Our goal is to provide a *labeling* for the pixel set X. This will be done by partitioning the graph into two disjoint sets(not X and L) by removing certain edges such that:

1 There is a many-to-one mapping from the set X to L. Every element of X must be mapped to one and only element of L.

2 The elements of L cannot belong to the same set. Thus there are no edges between elements of L and therefore the number of partitions is equal to the cardinality of L, and

3 The sum of the weights of the edges removed(the cut-set) is the minimum of all possible cut-sets.

In this example the cut-set is easily determined. For example, of the two edges connecting each element of X and an element of L, remove the edge with the *smaller* weight. Figure 5(b) shows the graph with the cut-set removed. Thus we have just shown that when the weights of the edges are interpreted as probabilities, the min-cut graph partition induces a

Maximum Apriori Estimate(MAP) estimate for the pixel labels. We prefer to say that the *min-cut induces a Bayesian classification* on the underlying pixel set. This is because we will use the Bayes theorem to calculate the edge weights of the graphs.

Example 2: Adding spatial constraints
In the previous example we did not use any information about the spatial proximity of the pixels relative to each other. We do that now by introducing additional edges in the graph structure.

Consider the graph shown in Figure 5(c) in which we have added two extra edges (x_1, x_2) and (x_2, x_3) with a weight λ. In this example we have chosen $\lambda = 0.2$.

Now if we want to retain the same partitions of the graph as in Example 1, then the cut-set has two extra edges, namely (x_1, x_2) and (x_2, x_3). Thus the sum of the weights of the edges in the cut-set $WC1$ is

$$WC1 = 0.3 + 0.4 + 0.45 + 2\lambda$$

But now, depending upon λ, the cut-set weight may not be minimal. For example, if $\lambda = 0.2$ then the weight of the cut-set $WC2$ consisting of the edges $\{(x_1, l_2), (x_2, l_1), (x_3, l_1), (x_1, x_2)\}$ is

$$WC2 = 0.3 + 0.4 + 0.55 + 0.2$$

Thus $WC2 < WC1$. What is happening is that if two neighboring pixels are assigned to different labels, then the edge between the two neighbors is added to the cut-set. Thus there is a penalty associated with two neighboring nodes being assigned every time to different labels. Thus we can model `spatial autocorrelation` by adding edges between the pixel nodes of the graph. We can also model `spatial heterogeneity` by assigning different *weights*, the λ's to the pixel edges.

3. Spatial Co-location Rules

Association rule finding [HGN00] is an important data mining technique which has helped retailers interested in finding items frequently bought together to make store arrangements, plan catalogs, and promote products together. In market basket data, a transaction consists of a collection of item types purchased together by a customer. Association rule mining algorithms [AS94, AS94] assume that a finite set of disjoint transactions are given as input to the algorithms. Algorithms like *apriori* [AS94] can efficiently find the frequent itemsets from all the transactions and association rules can be found from these frequent itemsets. Many spatial datasets consist of instances of a collection of boolean spatial features (e.g. drought, needle leaf vegetation). While boolean spatial features can be thought of as item types, there may not be an explicit finite set of transactions due to the continuity of underlying spaces. We define co-location rules, a generalization of association rules to spatial datasets, in this section.

3.1. Illustrative Application Domains

Many ecological datasets [LCM+97, NVA+99] consist of raster maps of the Earth at different times. Measurement values for a number of variables (e.g., temperature, pressure, and precipitation) are collected for different locations on Earth. Maps of these variables are available for different time periods ranging from twenty years to one hundred years. Some variables are measured using sensors while others are computed using model predictions.

A set of events, i.e., boolean spatial features, are defined on these spatial variables. Example events include drought, flood, fire, and smoke. Ecologists are interested in a variety of spatio-temporal patterns including co-location rules. Co-location patterns represent frequent co-occurrences of a subset of boolean spatial features. Examples of interesting co-location patterns in ecology are shown in Table 1.

Table 1. Examples of interesting spatio-temporal ecological patterns. Net Primary Production (NPP) is a key variable for understanding the global carbon cycle and the ecological dynamics of the Earth

Pattern #	Variable A	variable B	Examples of interesting patterns
P1	Cropland Area	Vegetation	Higher cropland area alters NPP
P2	Precipitation Drought Index	Vegetation	Low rainfall events lead to lower NPP
P3	Smoke Aerosol Index	Precipitation	Smoke aerosols alter the likelihood of rainfall in a nearby region
P4	Sea Surface Temperature	Land Surface Climate and NPP	Surface ocean heating affects regional terrestrial climate and NPP

The spatial patterns of ecosystem data sets include:

a. **Local co-location patterns** represent relationships among events in the same grid cell, ignoring the temporal aspects of the data. Examples from the ecosystem domain include patterns P1 and P2 of Table 1. These patterns can be discovered using algorithms [AS94] for mining classical association rules.

b. **Spatial co-location patterns** represent relationships among events happening in different and possibly nearby grid cells. Examples from the ecosystem domain include patterns P3 and P4 of Table 1.

Additional varieties of co-location patterns may exist. Furthermore, temporal natures of the ecosystem data give rise to many other time related patterns. We focus on the above co-location patterns in the following sections.

3.2. Co-location Rule Approaches

Given the difficulty in creating explicit disjoint transactions from continuous spatial data, this section defines several approaches to model co-location rules. We will use Figure 6 as an example spatial dataset to illustrate different models. In Figure 6, a uniform grid is imposed on the underlying spatial framework. For each grid l, its neighbors are defined to be the nine adjacent grids (including l). Spatial feature types are labeled beside their instances. We define following basic concepts to facilitate the description of different models.

Definition 1 *A* **co-location** *is a subset of boolean spatial features or spatial events.*

Definition 2 *A* **co-location rule** *is of the form:* $C_1 \rightarrow C_2(p, cp)$ *where* C_1 *and* C_2 *are co-locations,* p *is a number representing prevalence measure and* cp *is a number measuring conditional probability.*

The prevalence measure and the conditional probability measure are called interest measures and defined differently in different models which will be described shortly.

The **reference feature centric model** is relevant to application domains focusing on a specific boolean spatial feature, e.g. cancer. Domain scientists are interested in finding the co-locations of other task relevant features (e.g. asbestos, other substances) to the reference feature. This model enumerates neighborhoods to "materialize" a set of transactions around instances of the reference spatial feature. A specific example is provided by the spatial association rule [KH95].

For example, in Figure 6 a), let the reference feature be A, the set of task relevant features be B and C, and the set of spatial predicates include one predicate named "*close_to*". Let us define *close_to*(a, b) to be true if and only if b is a's neighbor. Then for each instance of spatial feature A, a transaction which is a subset of relevant features $\{B, C\}$ is defined. For example, for the instance of A at (2,3), transaction $\{B, C\}$ is defined because the instance of B at (1,4) (and at (3,4)) and instance

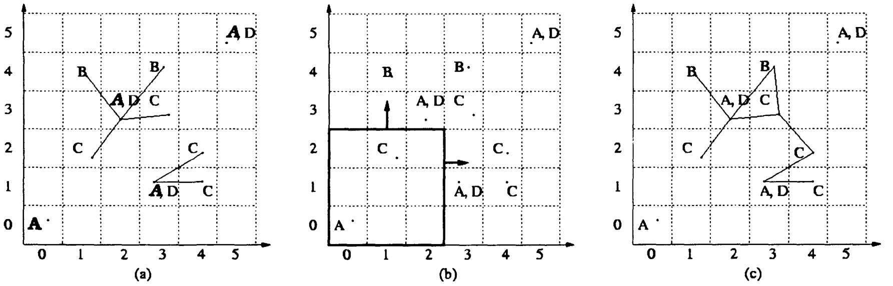


Figure 6. Spatial dataset to illustrate different co-location models. Spatial feature types are labeled besides their instances. The 9 adjacent grids of a grid l (including l) are defined to be l's neighbors. a) Reference feature centric model. The instances of A are connected with their neighboring instances of B and C by edges. b) Window centric model. Each 3 X 3 window corresponds to a transaction. c) Event centric model. Neighboring instances are joined by edges.

of C at (1,2) (and at (3,3)) are *close_to* (2,3). The transactions defined around instances of feature A are summarized in Table 2.

Table 2. Reference feature centric view: transactions are defined around instances of feature A relevant to B and C in figure 6 a)

Instance of A	Transaction
(0,0)	$\emptyset$
(2,3)	$\{B, C\}$
(3,1)	$\{C\}$
(5,5)	$\emptyset$

With "materialized" transactions, the support and confidence of the traditional association rule problem [AS94] may be used as prevalence and conditional probability measures as summarized in Table 3. Since 1 out of 2 non-empty transactions contains instances of both B and C and 1 out of 2 non-empty transactions contain C in Table 2, an association rule example is: $is\_type(i, A) \wedge \exists j\, is\_type(j, B) \wedge close\_to(j, i) \rightarrow \exists k\, is\_type(k, C) \wedge close\_to(k, i)$ with $\frac{1}{1} * 100\% = 100\%$ probability.

The **window centric model** is relevant to applications like mining, surveying and geology, which focus on land-parcels. A goal is to predict sets of spatial features likely to be discovered in a land parcel given that some other features have been found there. The window centric model enumerates all possible windows as transactions. In a space discretized by a uniform grid, windows of size kXk can be enumerated and materialized, ignoring the boundary effect. Each transaction contains a subset of spatial features of which at least one instance occurs in the corresponding window. The support and confidence of the traditional association rule problem may again be used as prevalence and conditional probability measures as summarized in Table 3. There are 16 3X3 windows corresponding to 16 transactions in Figure 6 b). All of them contain A and 15 of them contain both A and B. An example of an association rule of this model is: $an\, instance\, of\, type\, A\, in\, a\, window \rightarrow an\, instance\, of\, type\, B\, in\, this\, window$ with $\frac{15}{16} = 93.75\%$ probability. A special case of the window centric model relates to the case when windows are spatially disjoint and form a partition of space. This case is relevant when analyzing spatial datasets related to the units of political or administrative boundaries (e.g. country, state, zip-code). In some sense this is a local model since we treat each arbitrary partition as a transaction to derive co-location patterns without considering any patterns cross partition boundaries. The window centric model "materializes" transactions in a different way from the reference feature centric model.

The **event centric model** is relevant to applications like ecology where there are many types of boolean spatial features. Ecologists are interested in finding subsets of spatial features likely to occur in a neighborhood around instances of given subsets of event types. For example, let us determine the probability of finding at least one instance of feature type B in the neighborhood of an instance of feature type A in Figure 6 c). There are four instances of type A and only one of them have some instance(s) of type B in their 9-neighbor adjacent neighborhoods. The conditional probability for the co-location rule is: $spatial\, feature\, A\, at\, location\, l \rightarrow spatial\, feature\, type\, B\, in\, 9 - neighbor\, neighborhood\, is\, 25\%$.

Neighborhood is an important concept in the event centric model. Given a reflexive and symmetric neighbor relation R, we can define neighborhoods of a location l which satisfies the definition of neighborhood in Topology [Wor95] as follows:

Definition 3 *A* **neighborhood** *of l is a set of locations $L = \{l_1, \ldots, l_k\}$ such that l_i is a neighbor of l i.e. $(l, l_i) \in R(\forall i \in 1 \ldots k)$.*

We generalize the neighborhood definition to a collection of locations.

Definition 4 *For a subset of locations L' if L' is a neighborhood of every location in $L = \{l_1, \ldots, l_k\}$ then L' is a* **neighborhood** *of L.*

In another word, if every l_1 in L' is a neighbor of every l_2 in L, then L' is a neighborhood of L.

The definition of neighbor relation R is an input and is based on the semantics of application domains. It may be defined using topological relationships (e.g. connected, adjacent), metric relationships (e.g. Euclidean distance) or a combination (e.g. shortest-path distance in a graph such as roadmap). In general there are infinite neighborhoods over continuous space and it may not be possible to materialize all of them. But we are only interested in the locations where instances of spatial feature types (events) occurs. Even confined to these locations, enumerating all the neighborhoods incurs substantial computational cost because support based pruning cannot be carried out before the enumeration of all the neighborhoods is completed and the total number of neighborhoods is obtained. Furthermore, this support-based prevalence measure definition may not be meaningful because the value of the prevalences may be extremely small due to the fact that many neighborhoods are contained in bigger neighborhoods and counted multiple times. Thus the participation index is proposed to be a prevalence measure as defined below.

Definition 5 *For a co-location* $C = \{f_1, \dots, f_k\}$ *and a set of locations* $I = \{i_1, \dots, i_k\}$ *where* i_j *is an instance of feature* $f_j (\forall j \in 1, \dots, k)$ *if* I *is a neighborhood of* I *itself then* I *is an* **instance** *of* C.

In other words, if elements of I are neighbors to each other, then I is an instance of C. For example, {(3,1),(4,1)} is an instance of co-location $\{A, C\}$ in Figure 6 c) using a 9-neighbor adjacent neighbor definition.

Definition 6 *The* **participation ratio** $pr(C, f_i)$ *for feature type* f_i *of a co-location* $C = \{f_1, f_2, \dots, f_k\}$ *is the fraction of instances of* f_i *which participate in the co-location* C. *It can be formally defined as* $\frac{|distinct(\pi_{f_i}(all\ instances\ of\ co{-}location\ C))|}{|instances\ of\ \{f_i\}|}$ *where* π *is a projection operation.*

For example, in Figure 6 c), instances of co-location $\{A, B\}$ are {(2,3), (1,4)} and {(2,3)), (3,4)}. Only one instance (2,3) of spatial feature A out of four participates in co-location $\{A, B\}$. So $pr(\{A, B\}, A) = \frac{1}{4} = .25$.

Definition 7 *The* **participation index** *of a co-location* $C = \{f_1, f_2, \dots, f_k\}$ *is* $\prod_{i=1}^{k} pr(C, f_i)$.

In Figure 6 c), participation ratio pr($\{A, B\}, A$) of feature A in co-location $\{A, B\}$ is .25 as calculated above. Similarly $pr(\{A, B\}, B)$ is 1.0. The participation index for co-location $\{A, B\}$ is .25 X 1.0 = .25.

The conditional probability of a co-location rule $C_1 \rightarrow C_2$ in the event centric model is the probability of finding C_2 in a neighborhood of C_1 or it can be formally defined as:

Definition 8 *The conditional probability of a co-location rule* $C_1 \rightarrow C_2$ *is* $\frac{|distinct(\pi_{C_1}(all\ instances\ of\ co\text{-}location\ C_1 \cup C_2))|}{|instances\ of\ C_1|}$ *where* π *is a projection operation.*

For details of algorithms which mine co-location rules in event centric model refer to [SH01].

Table 3. Interest measures for different models

Model	Items	transactions defined by	Interest measures for $C_1 \rightarrow C_2$	
			Prevalence	Conditional probability
local	boolean feature types	partitions of space	fraction of partitions with $C_1 \cup C_2$	$Pr(C_2$ in a partition given C_1 in the partition)
reference feature centric	predicates on reference and relevant features	instances of reference feature C_1 and C_2 involved with	fraction of instance of reference feature with $C_1 \cup C_2$	$Pr(C_2$ is true for an instance of reference features given C_1 is true for that instance of reference feature)
window centric	boolean feature types	possibly infinite set of distinct overlapping windows	fraction of windows with $C_1 \cup C_2$	$Pr(C_2$ in a window given C_1 in that window)
event centric	boolean feature types	neighborhoods of instances of feature types	participation index of $C_1 \cup C_2$	$Pr(C_2$ in a neighborhood of $C_1)$

4. Spatial Outliers

Outliers have been informally defined as observations which appear to be inconsistent with the remainder of that set of data [BL94], or which deviate so much from other observations as to arouse suspicions that they were generated by a different mechanism [Haw80]. Spatial outliers are observations that are inconsistent with those in their neighbhorhood, even though they may not be inconsistent with the overall population. The identification of spatial outliers can lead to the discovery of unexpected knowledge and has a number of practical applications in areas related to transportation, epidemiology, precision agriculture, natural resource management, voting irregularity, and weather prediction.

Many outlier detection algorithms have been recently proposed; however, spatial outlier detection remains challenging for some reasons.

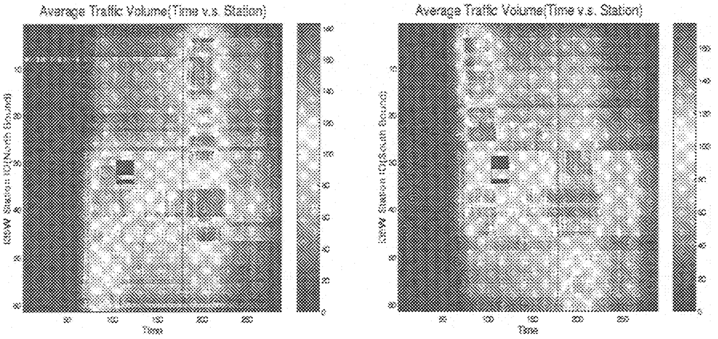
First, the choice of a neighborhood is non-trivial. Second, the design of statistical tests for the spatial outliers needs to account for the distribution of the attribute values at various locations as well as the distribution of aggregate function of attribute values over the neighborhoods. In addition, the computational cost of determining parameters for a neighborhood-based test can be high due to the possibility of join computations

4.1. An Illustrative Application Domain: Traffic Data Set

In 1995, the University of Minnesota and the Traffic Management Center(TMC) Freeway Operations group started the development of a database to archive sensor network measurements from the freeway system in the Twin Cities. The sensor network includes about nine hundred stations, each of which contains one to four loop detectors, depending on the number of lanes. Sensors embedded in the freeways and interstate monitor the occupancy and volume of traffic on the road. At regular intervals, this information is sent to the Traffic Management Center for operational purposes, e.g., ramp meter control, as well as research on traffic modeling and experiments.

In this application, we are interested in discovering the location of stations whose measurements are inconsistent with those of their spatial neighbors and the time periods when those abnormalities arise. The outlier detection tasks are: a) Build a statistical model for a spatial dataset; b) Check whether a specific station is an outlier; c) Check whether stations on a route are outliers.

Figure 7 shows an example of traffic flow outliers. Figure 7(a) and (b) are the traffic volume maps for I-35W North Bound and South Bound, respectively, on January 21 1997. The X-axis is the 5-minute time slot for the whole day and the Y-axis is the label of the stations installed on the highway, starting from 1 in the north end to 61 in the south end. The abnormal dark line at time slot 177 and the dark rectangle during time slot 100 to 120 on the X-axis and between station 29 to 34 on the Y-axis can be easily observed from both (a) and (b). This dark line at time slot 177 is an instance of temporal outliers, where the dark rectangle is a spatial-temporal outlier. Moreover, station 9 in Figure 7(a) exhibits inconsistent traffic flow compared with its neighboring stations, and was marked as a spatial outlier.



(a) I-35W North Bound (b) I-35W South Bound

Figure 7. An example of an outlier

4.2. Spatial Outlier Detection Approaches

Outliers in a spatial data set can be classified into three categories, namely, set-based outliers, multi-dimensional space-based outliers, and graph-based outliers. A set-based outlier is a data object whose attributes are inconsistent with attribute values of other objects in a given data set regardless of spatial relationships. Both multi-dimensional space-based outliers and graph-based outliers are called spatial outliers, that is, data objects that are significantly different in the attribute space from the collection of data objects among spatial neighborhoods. However, multi-dimensional space-based outliers and graph-based outliers are based on different spatial neighborhood definitions. In multi-dimensional space-based outlier detection, the definition of spatial neighborhood is based on Euclidean distance, while in graph-based spatial outlier detection, the definition is based on graph connectivity.

Many outlier detection algorithms [ABKS99, BL94, BKNS99, KN97, KN98, PS88, RR96, YSZ99] have been recently proposed, as shown in Figure 8. The set-based outlier detection algorithms [BL94, Joh92] consider the statistical distribution of attribute values, ignoring the spatial relationships among data objects. Numerous distribution-based outlier detection tests, known as discordancy tests [BL94, Joh92], have been developed for different circumstances, depending on the data distribution, the number of expected outliers, and the types of expected outliers. The main idea is to fit the data set to a known standard distribution, and develop a test based on distribution properties.

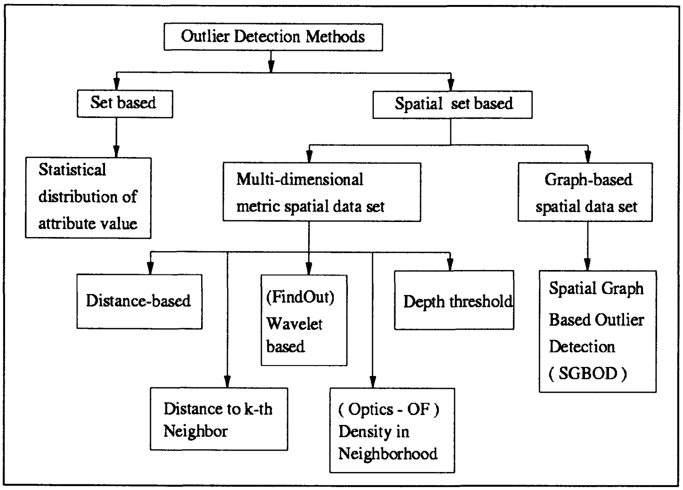

Outlier Detection Methods
Set based
Spatial set based
Statistical distribution of attribute value
Multi-dimensional metric spatial data set
Graph-based spatial data set
Distance-based
(FindOut) Wavelet based
Depth threshold
Spatial Graph Based Outlier Detection (SGBOD)
Distance to k-th Neighbor
(Optics - OF) Density in Neighborhood

Figure 8. Classification of outlier detection methods

The multi-dimensional space-based methods model data sets as a collection of points in a multidimensional space, and provide tests based on concepts such as distance, density, and convex-hull depth. Knorr and Ng presented the notion of distance-based outliers [KN97, KN98]. For a k dimensional data set T with N objects, an object O in T is a $DB(p, D)$-outlier if at least a fraction p of the objects in T lies greater than distance D from O. Ramaswamy et al. [RRS] proposed a formulation for distance-based outliers based on the distance of a point from its k^{th} nearest neighbor. After ranking points by the distance of each point to its k^{th} nearest neighbor, the top n points are declared as outliers. Breunig et al. [BKNS99] introduced the notion of a "local" outlier where the outlier-degree of an object is determined by taking into account the clustering structure in a bounded neighborhood of the object, e.g., k nearest neighbors. They formally defined the *outlier factor* to capture this relative degree of isolation or outlierness. Their notions of outliers are based on the same theoretical foundation as density-based cluster analysis [ABKS99]. In computational geometry, some depth-based approaches [RR96, PS88] organize data objects in convex hull layers in data space according to peeling depth [PS88], and outliers are expected to be found from data objects with shallow depth value. Conceptually, depth-based outlier detection methods are capable of processing multidimensional datasets. However, with the best case computational complexity $\Omega(N^{\lceil k/2 \rceil})$ for computing a convex hull, where N is the number of objects and k is the dimensionality of the dataset, depth-based outlier detection methods may not be applicable for high dimensional data sets. Yu et al. [YSZ99] introduced an outlier detection approach, called *FindOut*, which identifies outliers by removing clusters from the original data. Its key idea is to apply signal processing techniques to transform the space and find the dense regions in the transformed space. The remaining objects in the non-dense regions are labeled as outliers.

Multi-dimensional Euclidean spatial based methods detect outliers in multidimensional data space. These approaches have some limitations. First, the multi-dimensional approaches assume that the data items are embedded in isometric metric space and do not capture the spatial graph structure. Consider the application domain of traffic data analysis. A multi-dimensional method may put a detector station in the neighborhood of another detector even if the detectors were on opposite sides of the highway (e.g., I-35W north bound at exit 230, and I-35W south bound at exit 230), leading to the potentially incorrect identification of a bad detector. Secondly, these methods do not exploit apriori information about the statistical distribution of attribute data. Finally, they seldom provide the confidence measure of the discovered outliers.

In this following subsection, we describe a general framework for detecting spatial outliers in a spatial data set with an underlying graph structure. The detailed work can be found in [SLZ01].

Choice of Spatial Statistic. For spatial statistics, several parameters should be pre-determined before running the spatial outlier test. First, the neighborhood must be selected, based on a fixed cardinality or a fixed graph distance or a fixed Euclidean distance. Second, the aggregate neighborhood function must be chosen, e.g., mean, variance, and auto-correlation. The third parameter that should be pre-determined is the values used for comparing a location with its neighbors, either just one attribute or a vector of attribute values. Finally, the statistic must be chosen.

The statistic used in the graph-based outlier detection method is $S(x) = [f(x) - E_{y\in N(x)}(f(y))]$, where $f(x)$ is the attribute value for a data record x, $N(x)$ is the fixed cardinality set of neighbors of x, and $E_{y\in N(x)}(f(y))$ is the average attribute value for neighbors of x. Statistic $S(x)$ denotes the difference of attribute value of each data object x and the average attribute value of $x's$ neighbors.

Characterizing the Distribution of the Statistic.

Lemma 1 *Spatial Statistic* $S(x) = [f(x) - E_{y\in N(x)}(f(y))]$ *is normally distributed if attribute value* $f(x)$ *is normally distributed.*

Proof:

Given the definition of neighborhood, for each data record x, the average attribute values $E_{y \in N(x)}(f(y))$ of $x's$ k neighbors can be calculated. Since attribute values $f(x)$ are normally distributed and an average of normal variables is also normally distributed, the average attribute values $E_{y \in N(x)}(f(y))$ over neighborers is also a normal distribution for a fixed k cardinality neighborhood.

Since the attribute value and the average attribute value over neighbors are two normal variables, the distribution of difference $S(x)$ of each data object x and the average attribute value of $x's$ neighbors is also normally distributed.

Test for Outlier Detection. The test for detecting an outlier in a graph structure can be described as follows. $|\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. For each data object x with an attribute value $f(x)$, the $S(x)$ is the difference of the attribute value of data object x and the average attribute value of its neighbors. μ_s is the mean value of all $S(x)$, and σ_s is the standard deviation of all $S(x)$. The choice of θ depends on the specified confidence interval. For example, a confidence interval of 95 percent will lead to $\theta \approx 2$.

4.3. Computation of Test Parameters

We now describe the Test Parameters Computation(*TPC*) algorithm to calculate the test parameters, e.g., mean and standard deviation for the statistics. The computed mean and standard deviation can then be used to validate the outlier of the incoming data set.

Given an attribute data set V and the connectivity graph $G = (V, E)$, the *TPC* algorithm first retrieves the neighbor nodes from G for each data object x, then it computes the difference of the attribute value of x to the average of the attribute values of $x's$ neighbor nodes. These different values are then stored as a set called the AvgDist_Set. Finally, the AvgDist_Set is used to get the distribution value μ_s and σ_s.

4.4. Computation of Test Results

The neighborhood aggregate statistics value, e.g., mean and standard deviation, computed in the *TPC* algorithm can be used to verify the outlier of an incoming data set. The two verification procedures are Route Outlier Detection(*ROD*) and Random Node Verification(*RNV*). The *ROD* procedure detects the spatial outliers from a user specified route with the graph, e.g., all stations along a highway. The RNV procedure check the outlierness from a set of randomly generated nodes. The steps to detect outliers in both ROD and RNV are similar

Given a route RN in the data set V with graph structure $G = (V, E)$, the ROD algorithm first retrieves the neighboring nodes from G for each data object x in the route RN, then it computes the difference $S(x)$ between the attribute value of x and the average of attribute values of $x's$ neighboring nodes. Each $S(x)$ can then be tested using the spatial outlier detection test $|\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. The θ is pre-determined by the given confidence interval. The data objects with a statistic greater than θ are then declared as outliers.

5. Conclusion

In this chapter, we provide a new viewpoint for understanding the spatial data mining literature. Two major approaches are identified: the classical data mining method after feature selection and new spatial data mining approaches. We focused on the second approach. We introduced the spatial autoregression model and Markov random fields with graph partitioning to remove trends resulting from spatial dependency in classical location prediction models. A new pattern, co-location rules mining, which is associated with classical association rules mining, but different due to the intrinsic implicit transactions in spatial data, is introduced along with new prevalence measures and conditional probability definitions. Spatial outlier detection models spatial dependence via neighborhood graphs. Finally, there are other interesting spatial patterns including hotspot analysis, aggregate proximity [KN96, KR96], and boundary shape matching [KNS97].

Acknowledgments

We thank Jiawei Han (Simon Fraser University, CA), James Lesage, and Sucharita Gopal(Boston University) for valuable insights during the discussion on spatial data mining at the Scientific Data Mining workshop 2000 held at the Army High Performance Computing Research Center. We also thank Ugyar Ozesmi (Ericyes University, Kayseri, Turkey), and Pusheng Zhang (University of Minnesota) for contributions of dataset and graphs and Xiaobin Ma and Hui Xiong for their valuable feedback on early versions of this paper. We would like to thank Steve Klooster et. al. from NASA for the examples in Table 1, and James Lesage (http://www.spatial-econometrics.com/) for making the MATLAB toolbox available on the web.

References

[ABKS99] M. Ankerst, M.M. Breunig, H.P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, USA*, pages 49–60, 1999.

[Agr94] R. Agrawal. Tutorial on database mining. In *Thirteenth ACM Symposium on Principles of Databases Systems*, pages 75–76, Minneapolis, MN, 1994.

[AM95] P.S. Albert and L.M. McShane. A Generalized Estimating Equations Approach for Spatially Correlated Binary Data: Applications to the Analysis of Neuroimaging Data. *Biometrics(Publisher: Washington, Biometric Society, Etc.), 1:627-638*, 1995.

[Ans88] L Anselin. *Spatial Econometrics: methods and models.* Kluwer, Dordrecht, Netherlands, 1988.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for Mining Association Rules. In *Proc. of Very Large Databases*, may 1994.

[BKNS99] M.M. Breunig, H.P. Kriegel, R. T. Ng, and J. Sander. Optics-of: Identifying local outliers. In *Proc. of PKDD '99, Prague, Czech Republic, Lecture Notes in Computer Science (LNAI 1704), pp. 262-270, Springer Verlag*, 1999.

[BL94] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley, New York, 3rd edition, 1994.

[BVZ99] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts . *Proc. of International Conference on Computer Vision*, September 1999.

[Cre93] N.A. Cressie. *Statistics for Spatial Data (Revised Edition).* Wiley, New York, 1993.

[EKS97] M. Ester, H.-P. Kriegel, and J. Sander. Spatial Data Mining: A Database Approach. In *Proc. Fifth Symposium on Rules in Geographic Information Databases*, 1997.

[Gre00] G. Greenman. Turning a map into a cake layer of information. *New York Times, http://www.nytimes.com/library/tech /00/01/circuits/articles/20giss.html*, Feb 12 2000.

[Gut94] R.H. Guting. An Introduction to Spatial Database Systems. In *Very Large Data Bases Journal(Publisher: Springer Verlag)*, October 1994.

[Hai89] R.J. Haining. Spatial Data Analysis in the Social and Environmental Sciences. In *Cambridge University Press, Cambridge, U.K*, 1989.

[Haw80] D. Hawkins. *Identification of Outliers.* Chapman and Hall, 1980.

[HGL93] M. Hohn, L. Gribki, and A.E. Liebhold. A Geostatistical Model for Forecasting the Spatial Dynamics of Defoliation Caused by the Beypsy MothLymantria dispar (Lepidoptera: Lymantriidae). *Environmental Entomology (Publisher: Entomological Society of America, 22:1066-1075*, 1993.

[HGN00] J. Hipp, U. Guntzer, and G. Nakaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[IES89] Issaks, Edward, and M. Svivastava. Applied Geostatistics. In *Oxford University Press, Oxford*, 1989.

[Joh92] R. Johnson. *Applied Multivariate Statistical Analysis.* Prentice Hall, 1992.

[KAH96] K. Koperski, J. Adhikary, and J. Han. Knowledge Discovery in Spatial Databases: Progress and Challenges. In *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Montreal, Canada. 55-70*, 1996.

[KH95] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Proc. Fourth International Symposium on Large Spatial Databases, Maine. 47-66*, 1995.

[KHS98] K. Koperski, J. Han, and N. Stefanovic. An Efficient Two-Step Method for Classification of Spatial Data. 1998.

[KN96] E.M. Knorr and R.T. Ng. Extraction of Spatial Proximity Patterns by Concept Generalization. In *Proc. Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon. AAAI Press. 347-350*, 1996.

[KN97] E. Knorr and R. Ng. A unified notion of outliers: Properties and computation. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 219–222, 1997.

[KN98] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. 24th VLDB Conference*, 1998.

[KNS97] E.M. Knorr, R.T. Ng, and D.L. Shilvock. Finding boundary shape matching relationships in spatial data. In *Proc. 5th International Symposium, SSD 97. Springer-Verlag, Berlin, 29-46*, 1997.

[KR96] E.M. Knorr and Ng R.T. Finding aggregate proximity relationships and commonalities in spatial data mining. *IEEE Trans. Knowl. and Data Eng. 8(6):884-897*, 1996.

[Kru95] P. Krugman. Development, Geography, and Economic theory. In *MIT Press, Cambridge, MA*, 1995.

[LCM+97] Z. Li, J. Cihlar, L. Moreau, F. Huang, and B. Lee. Monitoring Fire Activities in the Boreal Ecosystem. *Journal Geophys. Res,. 102(29):611-629*, 1997.

[LeS97] J.P. LeSage. Bayesian estimation of spatial autoregressive models. *International Regional Science Review*, (20):113–129, 1997.

[Mar99] D. Mark. Geographical Information Science: Critical Issues in an Emerging Cross-disciplinary Research Domain. In *NSF Workshop*, February 1999.

[NVA+99] D.C. Nepstad, A. Verissimo, A. Alencar, C. Nobre, E. Lima, P. Lefebvre, P. Schlesinger, C. Potter, P. Moutinho, E. Mendoza, M. Cochrane, and V. Brooks. Large-scale Impoverishment of Amazonian Forests by Logging and Fire. *Nature, 398:505-508*, 1999.

[OM97] U. Ozesmi and W. Mitsch. A spatial habitat model for the Marsh-breeding red-winged black-bird(agelaius phoeniceus l.) In coastal lake Erie wetlands. *Ecological Modelling (Publisher: Elsevier Science B. V.)*, (101):139–152, 1997.

[OO99] S. Ozesmi and U. Ozesmi. An Artificial neural network approach to spatial habitat modeling with interspecific interaction. *Ecological Modelling (Publisher: Elsevier Science B. V.)*, (116):15–31, 1999.

[PS88] F. Preparata and M. Shamos. *Computatinal Geometry: An Introduction.* Springer Verlag, 1988.

[Qui86] J.R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1986.

[RR96] I. Ruts and P. Rousseeuw. Computing depth contours of bivariate point clouds. In *Computational Statistics and Data Analysis, 23:153–168*, 1996.

[RRS] S. Ramaswamy, R. Rastongi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Bell Laboratories, Murray Hill, NJ.*

[RS99] J.-F. Roddick and M. Spiliopoulou. A Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research. *SIGKDD Explorations 1(1): 34-38 (1999)*, 1999.

[SC01] S. Shekhar and S. Chawla. Spatial Databases: Issues, Implementation and Trends. *Prentice Hall (under contract)*, 2001.

[SCR$^+$99] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.-T. Lu. Spatial Databases - Accomplishments and Research Needs. *Trans. on Knowledge and Data Engineering 11(1): 45-55 (1999)*, 1999.

[SH01] S. Shekhar and Y. Huang. Co-location Rules Mining: A Summary of Results. *Proc. Spatio-temporal Symposium on Databases*, 2001.

[SLZ01] S. Shekhar, C.T. Lu, and P Zhang. Detecting Graph-based Spatial Outliers: Algorithms and Applications . In *Department of Computer Science Techinical Report TR 01-014, University of Minnesota: http://tiberius.cs.umn.edu/techreports/listing/*, 2001.

[SNM$^+$95] P. Stolorz, H. Nakamura, E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng, S.Y. Chien, R. Mechoso, and J.D. Farrara. Fast Spatio-Temporal Data Mining of Large Geophysical Datasets. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, AAAI Press, 300-305*, 1995.

[SYH93] S. Shekhar, T.A. Yang, and P. Hancock. An Intelligent Vehicle Highway Information Management System. *Intl Jr. on Microcomputers in Civil Engineering (Publisher: Blackwell Publishers), 8(3)*, 1993.

[Tob79] W.R. Tobler. *Cellular Geography, Philosophy in Geography.* Gale and Olsson, Eds., Dordrecht, Reidel, 1979.

[Wor95] M.F. Worboys. *GIS A Computing Perspective.* Taylor and Francis, 1995.

[YL97] Y. Yasui and S.R. Lele. A Regression Method for Spatial Disease Rates: An Estimating Function Approach. *Journal of the American Statistical Association, 94:21-32*, 1997.

[YSZ99] D. Yu, G. Sheikholeslami, and A. Zhang. Findout: Finding outliers in very large datasets. In *Depart-*

ment of Computer Science and Engineering State University of New York at Buffalo, Technical report 99-03, http://www.cse.buffalo.edu/tech-reports/, 1999.

Chapter 27

PREDICTING FAILURES IN EVENT SEQUENCES

Mohammed J. Zaki, Neal Lesh, and Mitsunori Ogihara

Abstract In this paper we develop new techniques for predicting failures and monitoring in categorical event sequences. New techniques are needed because failures are rare and previous data mining algorithms were overwhelmed by the staggering number of very frequent, but entirely unpredictive patterns that exist in such databases. This paper combines several techniques for pruning out unpredictive and redundant patterns, which reduce the size of the returned rule set by more than three orders of magnitude. As a concrete application, we present PlanMine, an algorithm to extract patterns of events that predict failures in databases of plan executions. PLANMINE has also been fully integrated into two real-world planning systems. We experimentally evaluate the rules discovered by PLANMINE, and show that they are extremely useful for understanding and improving plans, as well as for building monitors that raise alarms before failures happen.

Keywords: Data Mining, Sequential Patterns, Association Rules, Predicting Failures, Mining Rare Events

1. Introduction

Sequences abound in all kinds of categorical or "symbolic" data. Examples of such sequences include biological sequences of DNA or Amino Acids, web access logs as well as text and hypertext documents, network alarms, plan execution traces, multi-player games, simulation data, etc.

One of the important kind of mining that can be performed on such categorical sequences is that of classification. Usually, the classification task is binary with a highly skewed class distribution. For example, given network logs, a secure site might want to know if there have been any unauthorized accesses. This is the intrusion detection problem. The vast majority of traces belong to the "normal" class, while only a very

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 515–539.
© 2001 *Kluwer Academic Publishers.*

small fraction of all logs might be "intrusion." As another compelling example, consider a plan execution database for important problems like resume missions or forest-fire prevention, etc. Here we would like to know if a plan is likely to "succeed" or to "fail". We expect that most of the events and plans are successful, and while failures are very rare, the cost of a failure is high. Thus the aim of mining would be to extract high confidence rules that predict failures and to even build monitors that might signal a failure before it happens so that preventive steps can be taken.

In this paper, we present the PLANMINE algorithm for mining such failure information from plan execution traces. We apply sequence data mining to extract causes of plan failures, and feed the discovered patterns back into the planner to improve future plans. We also use the mined rules for building monitors that signal an alarm before a failure is likely to happen. While the techniques we present are in the context of planning, we would like to underscore that our approach is equally applicable in other highly skewed sequence classification problems such as intrusion detection, or finding rare frequent patterns in DNA sequences, etc.

We show that one cannot simply apply previous sequence discovery algorithms [SA96, Zak98] for mining execution traces. Due to the complicated structure and redundancy in the data, simple application of the known algorithms generates an enormous number of highly frequent, but unpredictive rules. We use the following novel methodology for pruning the space of discovered sequences. We label each plan as "good" or "bad" depending on whether it achieved its goal or it failed to do so. Our goal is to find "interesting" sequences that have a high confidence of predicting plan failure. We developed a three-step pruning strategy for selecting only the most predictive rules:

1 *Pruning Normative Patterns*: We eliminate all *normative* rules that are consistent with background knowledge that corresponds to the normal operation of a (good) plan, i.e., we eliminate those patterns that not only occur in bad plans, but also occur in the good plans quite often, since these patterns are not likely to be predictive of bad events.

2 *Pruning Redundant Patterns*: We eliminate all *redundant* patterns that have the same frequency as at least one of their proper subsequences, i.e., we eliminate those patterns q that are obtained by augmenting an existing pattern p, while q has the same *frequency* as p. The intuition is that p is as predictive as q.

3 *Pruning Dominated Patterns*: We eliminate all *dominated* sequences that are less predictive than any of their proper subsequences, i.e., we eliminate those patterns q that are obtained by augmenting an existing pattern p, where p is shorter or more general than q, and has a higher *confidence* of predicting failure than q.

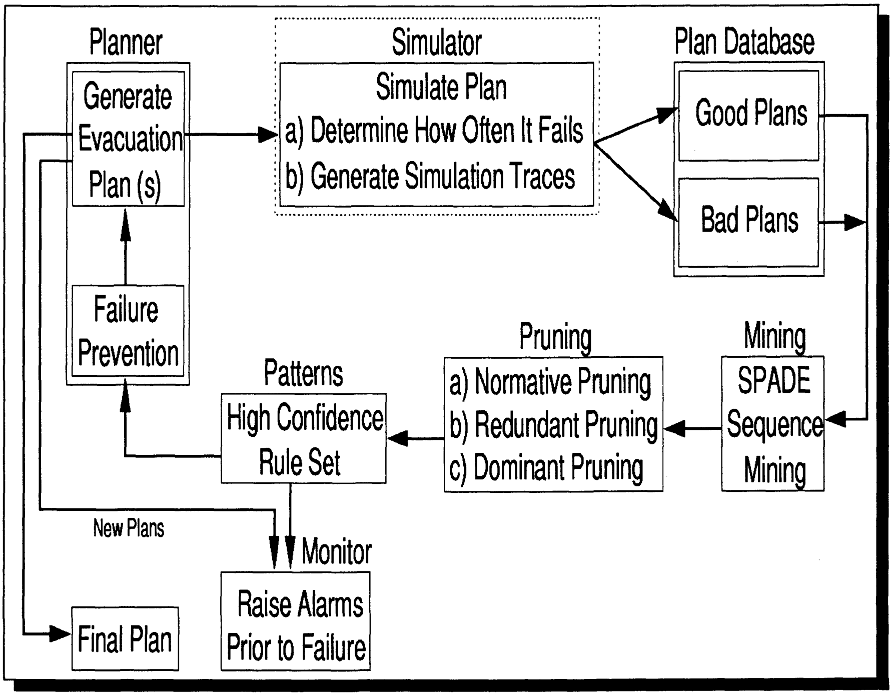

Planner
Generate Evacuation Plan (s)
Failure Prevention
Simulator
Simulate Plan
a) Determine How Often It Fails
b) Generate Simulation Traces
Plan Database
Good Plans
Bad Plans
Pruning
a) Normative Pruning
b) Redundant Pruning
c) Dominant Pruning
Mining
SPADE Sequence Mining
Patterns
High Confidence Rule Set
New Plans
Monitor
Raise Alarms Prior to Failure
Final Plan

Figure 1. PLANMINE Architecture

These three steps are carried out *automatically* by mining the good and bad plans separately and comparing the discovered rules from the unsuccessful plans against those from the successful plans. The complete architecture of PLANMINE is shown in Figure 1. There are two main goals: 1) to improve an existing plan, and 2) to generate a plan monitor for raising alarms. In the first case the planner generates a plan and simulates it multiple times. It then produces a database of good and bad plans in simulation. This information is fed into the mining engine, which discovers high frequency patterns in the bad plans. We next apply our pruning techniques to generate a final set of rules that are highly predictive of plan failure. This mined information is used for fixing the plan to prevent failures, and the loop is executed multiple times till no further improvement is obtained. The planner then generates the final plan. For the second goal, the planner generates multiple plans, and creates a database of good and bad plans (there is no simulation step). The high confidence patterns are mined as before, and the information is used to generate a plan monitor that raises alarms prior to failures in new plans.

PLANMINE has been integrated into two applications in planning: the TRIPS collaborative planning system [FJ98], and the IMPROVE algorithm for improving large, probabilistic plans [LMA98]. To experimentally validate our approach, we show that IMPROVE does not work well if the PLANMINE component is replaced by less sophisticated methods for choosing which parts of the plan to repair. We also show that the output of PLANMINE can be used to build execution monitors which predict failures in a plan before they occur. We were able to produce monitors with 100% precision, that signal 90% of all the failures that occur.

2. Related Work

There has been much research on analyzing planning episodes (i.e., invocations of the planner) to improve future planning performance in terms of efficiency or quality (e.g. [Min90]). Our work is quite different in that we are analyzing the performance of the plan, and not the planner.

In [McD94] a system is described in which a planning robot analyzes simulated execution traces of its current plan for bugs, or discrepancies between what was expected and what occurred. Each bug in the simulations is classified according to an extensive taxonomy of failure modes. This contrasts to our work in which we mine patterns of failure from databases of plans that contain many problems, some minor and some major, and the purpose of analysis is to discover important trends that distinguish plan failures from successes.

CHEF [Ham90] is a case-based planning system that also analyzes a simulated execution of a plan. CHEF simulates a plan once, and if the plan fails, applies a deep causal model to determine the cause of failure. This work assumes that the simulator has a correct model of the domain. [RT94] also described a method of producing execution monitors by analyzing the causal structure of the plan.

Additionally, there has been work on extending classical planners to probabilistic domains (e.g. [KHW95]). These methods have been applied to very small plans because the analytic techniques are exponential in the length of the plan. Furthermore, in this work, plan assessment and plan refinement are completely separate. They also mention the importance of using the results of assessing a probabilistic plan to help guide the choice of how to improve it. Currently, the probabilistic planner chooses randomly from among all the actions that *might* improve the plan. As shown in our second set of experiments, the patterns extracted by data mining can help focus a planner on what part of the plan to improve.

[HV98] applies machine learning techniques for robot path planning. Their system uses predictive features of the environment to create *situation-dependent costs* for arcs in a path map used by the planner to create routes for the robot. They use regression tree [BFOS84] for the mining engine, to learn separate models for each arc in the path. In our domain this would correspond to learning rules for each route in the evacuation domain. However, our goal is different in that we are trying to learn long sequences of events that cause plan failure.

In [HC95] a methodology called *dependency interpretation* is presented, that uses statistical dependency detection to identify interesting (unusually frequent or infrequent) patterns in plan execution traces. They then interpret the patterns using a weak model of the planner's interaction with its environment to explain how the patterns might have been caused by the planner. One limitation is that once the patterns have been detected, the user is responsible for picking up an interesting pattern from among the set of mined patterns, and using it for interpretation. In contrast, we automatically extract the set of the highly predictive patterns. They also applied the discovered patterns for preventing plan failures. However, they detect dependencies between a precursor and the failure that immediately follows it, and they found that they were likely to miss dependencies by not considering longer sequences. Our approach on the other hand detects long failure sequences. In their failure prevention study they only used traces of failure and recovery methods, and did not include other influences (e.g., changing weather). In contrast, we use all possible influences for discovering patterns. In [OC96], they applied the MSDD algorithm to detect rules among planning operators and the context. Our work contrasts with theirs in that in addition to detecting the frequent operators in the bad plans, we apply effective pruning techniques to automatically extract the rules most predictive of failure.

Our work is an important application of sequence mining [AS95, SA96, MTV97, Zak01]. It is built upon the SPADE algorithm [Zak98, Zak01], which was shown to outperform previous methods [SA96] by up to an order of magnitude. TASA [HKM$^+$96] is a related system which applies frequent episode mining to discover frequent alarms in telecommunication network databases.

The high item frequency in our domain distinguishes it from previous applications of sequential patterns. For example, while extracting patterns from mail order datasets [SA96], the database items had very low support, so that support values like 1% or 0.1% were used. For discovering frequent alarm sequences in telecommunication network alarm databases [HKM$^+$96] the support used was also 1% or less. Furthermore, in previous applications [HKM$^+$96] only a database of alarms (i.e., only the bad events) was considered, and the normal events of the network were ignored. In our approach, since we wanted to predict plan failures, we have shown that considering only the frequent sequences in the bad plans is not sufficient (all these have 100% confidence of predicting failure). We also had to use the good plans as a source of "normative" information, which was used to prune out unpredictive rules.

The problem formulation for predicting failures can be cast as discovering high confidence classification rules, where the class being predicted is whether a plan fails. However, there are several characteristics that render traditional classifiers ineffective in our domain. Firstly, we have a large number of attributes, many of which have missing values. Secondly, we are trying to predict rare events with very low frequency, resulting in skewed class distribution. Lastly, and most importantly, we are trying to predict sequence rules, while most previous work only targets non-temporal domains. The main difficulty is in finding good features to use to represent the plan traces for decision trees construction. One can find a reasonable set of features for describing an individual event (i.e., action in the plan) and then have a copy of this feature set by every time step. However, very soon we run into trouble, since our feature space of multiple event sequences can become exponential. A method to control this explosion is to bound the feature length to sequences of size 2 for example. However, this is likely to miss out on longer predictive rules.

An approach similar to ours, but applied in a non-temporal domain, is the partial classification work in [AMS97]. They try to predict rules for a specific class out or two or more classes. They isolate the examples belonging to a given class, mine frequent associations, and then compare the confidence based on the frequency for that class, and the frequency in the remaining database, which is similar to our background pruning. However, they don't do any redundant or dominant pruning, and as we mentioned above they do not consider sequences over time. A brute-force method for mining high-confidence classification rules using frequent associations was presented in [Bay97]. They also describes several pruning strategies to control combinatorial explosion in the number of candidates counted. One key difference is that we are working in the sequence domain, as opposed to the association domain. The Brute algorithm [RSE94] also performs a depth-bounded brute-force search for predictive rules, returning the k best ones. In one of their datasets applied to Boeing parts they do consider time, but their treatment is different. The dataset consists of part type and the time it spends at a particular workstation in a semi-automated factory. They treat time as another attribute, and the discovered rules may thus have a temporal component. In our data format each instance corresponds to a sequence of event sets. Time is not an attribute but is used to order the events, and we explicitly mine predictive sequences in this database.

3. Discovery of Plan Failures: Sequence Mining

Parameter	Value
Action	Move, Load, Unload
Outcome	Success, Tardy, Late, Overheat, Blowout, Flat, Crash
Route	Delta-Exodus, Exodus-Barnacle-Abyss, Delta-Calypso-Delta, ..
From	Abyss, Barnacle, Calypso, Delta, Exodus
To	Abyss, Barnacle, Calypso, Delta, Exodus
AtLocation	Abyss, Barnacle, Calypso, Delta, Exodus
Cargo	People7, Person1, Person2, etc.
Vehicle	Helicopter, Truck
VehicleId	Heli1, Truck1, Truck2
Weather	Good, Fair, Rough, Hazardous

Figure 2. Plan Database Parameters

The input to PLANMINE consists of a database of plans for evacuating people from one city to another. Each plan is tagged *Failure* or *Success* depending on whether or not it achieved its goal. Each plan has a unique identifier, and a sequence of actions or events. Each event is composed of several different items including the event time, the unique event identifier, the action name, the outcome of the event, and a set of additional parameters specifying the weather condition, vehicle type, origin and destination city, cargo type, etc. An example of the different parameter values is shown in Figure 2, and some example plans are shown in Figure 3.

While routing people from one city to another using different vehicles, the plan will occasionally run into trouble. The outcome of an event specifies the type of error that occurred, if any. Only a few of the errors, such as a helicopter crashing or a truck breaking down, are *severe*, and cause the plan to fail. However, a sequence of non-severe outcomes

PLAN DATABASE

PlanId	Time	EventId	Action	Outcome	Route	From	To	AtLocation	Cargo	Vehicle	VehicleId	Weather
1	10	78	Move	Success	Delta-Exodus	Delta	Exodus			Helicopter	Heli1	Good
1	20	84	Load	Success				Exodus	People7		Heli1	
1	30	85	Move	Flat	Exodus-Barnacle-Abyss	Exodus	Barnacle			Helicopter	Heli1	Fair
1	40	101	Unload	Crash				Barnacle	People7	Helicopter	Heli1	Hazardous
2	10	7	Move	Flat	Delta-Calypso-Delta	Delta	Calypso			Truck	Truck1	Good
2	20	10	Move	Breakdown	Delta-Calypso-Delta	Calypso	Delta			Truck	Truck1	Good

Figure 3. Example Plan Database

may also be the cause of a failure. For example, a rule might be *(Load People-7 Truck-1)* $\mapsto$ *(Move Flat Truck-1)* $\mapsto$ *(Move Late Truck-1)* $\mapsto$ *(Load People-7 Heli-1)* $\mapsto$ *(Move Crash Heli-1 RoughWeather)* $\Rightarrow$ *Failure*, indicating that the plan is likely to fail if *Truck-1* gets *Late* due to a *Flat*. This causes the *Helicopter-1* to crash, a severe outcome, since the weather gets *Rough* with time.

We now cast the problem of mining for causes of plan failures as the problem of finding *sequential patterns* [AS95]. Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of m distinct attributes, also called *items*. Each distinct parameter and value pair is an item. For example, in Figure 2, *Action=Move*, *Action=Load*, etc., are all distinct items (below, we omit the parameter name and refer to items only by their value). An *event* is an unordered collection of items, all of which are assumed to occur at the same time. Without loss of generality, we assume that the items are mapped to integers, and that items of an event are sorted in increasing order. An event i is denoted as $(i_1 i_2 \cdots i_k)$, where i_j is an item.

A *sequence* is an ordered list of events. A sequence α is denoted as $(\alpha_1 \mapsto \alpha_2 \mapsto \cdots \mapsto \alpha_n)$, where each sequence element α_j is an event. An item can occur only once in an event, but it can occur multiple times in different events of a sequence. A sequence with k items ($k = \sum_j |\alpha_j|$) is called a *k-sequence*. We say that α is a *subsequence* of β, denoted as

$\alpha \preceq \beta$, if there exist integers $i_1 < i_2 < \cdots < i_n$ such that $\alpha_j \subseteq \beta_{i_j}$ for all α_j. For example, $B \mapsto AC$ is a subsequence of $AB \mapsto E \mapsto ACD$. We say that α is a proper subsequence of β, denoted $\alpha \prec \beta$, if $\alpha \preceq \beta$ and $\beta \npreceq \alpha$. If α is obtained by removing a single item from β, we write it as $\alpha \prec_1 \beta$. For example, $(B \mapsto AC) \prec_1 (BE \mapsto AC)$.

We now cast our plans in the sequence mining framework. An event $\mathcal{E}$ has a unique identifier and is composed of different parameter values (the items). For example, in Figure 3 the second row (event) of the first plan is given by the item set *(84, Load, Success, Exodus, People7, Heli1)*. A *plan* or *plan-sequence* $\mathcal{S}$ has a unique identifier and is associated with a sequence of events $\mathcal{E}_1 \mapsto \mathcal{E}_2 \mapsto \cdots \mapsto \mathcal{E}_n$. Without loss of generality, we assume that no plan has more than one event with the same time-stamp, and that the events are sorted by the event-time. The input plan database, denoted $\mathcal{D}$, consists of a number of such plan-sequences.

Support. A plan $\mathcal{S}$ is said to *contain* a sequence α, if $\alpha \preceq \mathcal{S}$, i.e., if α is a subsequence of the plan-sequence $\mathcal{S}$. The *support* or *frequency* of a sequence α, denoted $fr(\alpha, \mathcal{D})$ is the fraction of plans in the database $\mathcal{D}$ that contain α, i.e.,

$$fr(\alpha, \mathcal{D}) = \frac{|\{\alpha \preceq \mathcal{S} \in \mathcal{D}\}|}{|\mathcal{D}|}.$$

According to this definition a sequence α is counted only once per plan even though it may occur multiple times in that plan. It is easy to modify this definition to count a sequence multiple times per plan, if the semantics of the problem require it. Given a user-specified threshold called the *minimum support* (denoted *min_sup*), we say that a sequence is *frequent* if $fr(\alpha, \mathcal{D}) \geq min\_sup$.

Confidence. Let α and β be two sequences. The *confidence* of a sequence rule $\alpha \Rightarrow \beta$ is the conditional probability that sequence β occurs, given that α occurs in a plan, given as

$$Conf(\alpha, \mathcal{D}) = \frac{fr(\alpha \mapsto \beta, \mathcal{D})}{fr(\alpha, \mathcal{D})}.$$

Given a user-specified threshold called the *minimum confidence* (denoted *min_conf*), we say that a sequence is *strong* if $Conf(\alpha, \mathcal{D}) \geq min\_conf$.

Discovery Task. Given a database $\mathcal{D}$ of good and bad plans, tagged as *Success* and *Failure*, respectively, the problem of discovering causes of plan failures can be formulated as finding all strong rules of the form $\alpha \Rightarrow$ *Failure*, where α is a frequent sequence. This task can be broken into two main steps:

1 Find all frequent sequences. This step is computationally and I/O intensive, since there can be potentially an exponential number of frequent sequences.

2 Generate all strong rules. Since we are interested in predicting failures, we only consider rules of the form $\alpha \Rightarrow$ *Failure*, although our formulation allows rules with consequents having multiple items. The rule generation step has relatively low computational cost.

We use the SPADE algorithm [Zak98] to efficiently enumerate all the frequent sequences. Generally a very large number of frequent patterns are discovered in the first step, and consequently a large number of strong rules are generated in the second step. If one thinks of the frequent sequence discovery step as the *quantitiative* step due to its high computational cost, then the rule generation step is the *qualitative* one, where the quality of the discovered rules is important and not the quantity. The main focus of this paper is on how to apply effective pruning techniques to reduce the final set of discovered rules, retaining only the rules that are most predictive of failure, and on how to do this automatically. In what follows, we usually omit the rule consequent i.e., *Failure*, since we mine only rules of the form $\alpha \Rightarrow$ *Failure*; the sequence α is used to refer to the entire rule.

4. Sequential Pattern Discovery Algorithm

We now briefly describe the SPADE [Zak98, Zak01] algorithm, that we used for efficient discovery of sequential patterns. SPADE is disk-based and is designed to work with very large datasets.

SPADE uses the observation that the subsequence relation $\preceq$ defines a partial order on the set of sequences, i.e., if β is a frequent sequence, then all subsequences $\alpha \preceq \beta$ are also frequent. The algorithm systematically searches the sequence lattice spanned by the subsequence relation, from the most general (single items) to the most specific frequent sequences (maximal sequences) in a depth-first (or breadth-first) manner. For example, let the set of frequent items $\mathcal{F}_1 = \{A, B, C\}$, and let the maximal frequent sequences be $(ABC) \mapsto A$, and $(B \mapsto A \mapsto C)$, then Figure 4 shows the lattice of frequent sequences induced by the maximal elements (note that a sequence is maximal if it is not a subsequence of any other sequence).

Given $\mathcal{F}_k$, the set of frequent sequences of length k, we say that two sequences belong to the same equivalence class if they share a common $k-1$ length prefix. For example, from the $\mathcal{F}_2$ shown in Figure 4, we obtain the following three equivalence classes: $[A] = \{A \mapsto A, A \mapsto C, AB, AC\}$; $[B] = \{B \mapsto A, B \mapsto C, BC\}$; and $[C] = \{C \mapsto A\}$. Each

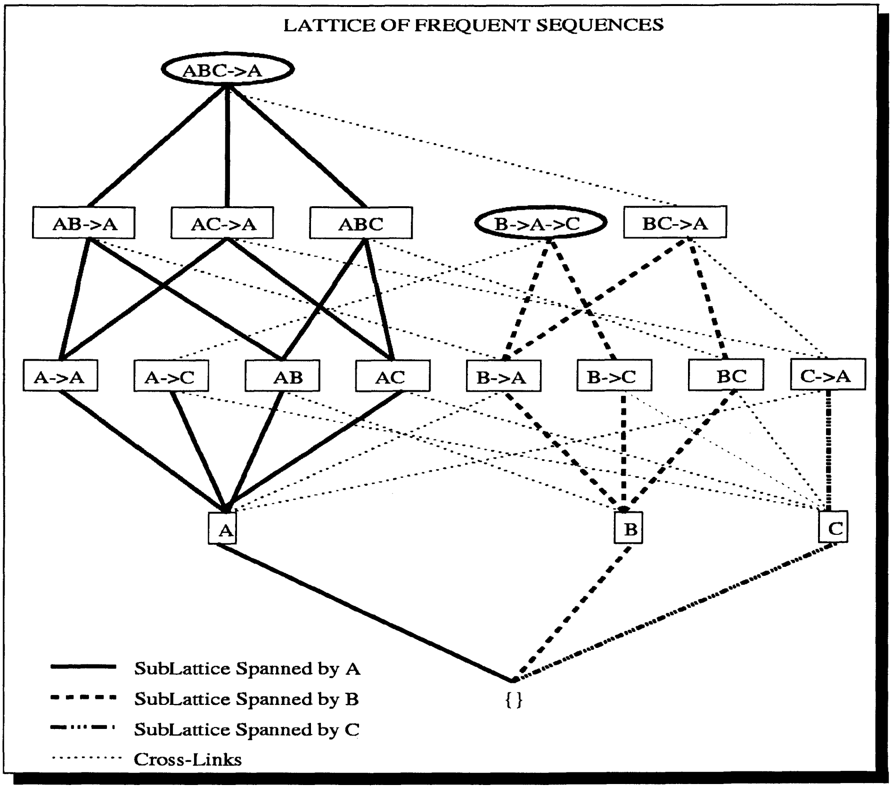

LATTICE OF FREQUENT SEQUENCES
ABC->A
AB->A
AC->A
ABC
B->A->C
BC->A
A->A
A->C
AB
AC
B->A
B->C
BC
C->A
A
B
C
{}
SubLattice Spanned by A
SubLattice Spanned by B
SubLattice Spanned by C
Cross-Links

Figure 4. Lattice Induced by Maximal Sequences

class [℘] has complete information for generating all frequent sequences with the prefix ℘. Each class can thus be solved independently.

SPADE decomposes the original problem into smaller subproblems by the recursive partitioning into equivalence classes (sub-lattices). This allows each class to be processed entirely in main-memory, and generally requires up to two complete database scans. Figure 5 shows the outline of the algorithm. SPADE systematically searches the sub-lattices in a breadth-first manner (SPADE can also use a depth-first search if main-memory is limited), i.e. it starts with the frequent single items, and during each iteration the frequent sequences of the previous level are extended by one more item. Before computing the support of a newly formed sequence, a check is made to ensure that all its subsequences are frequent. If any subsequence is found to be infrequent, then the sequence cannot possibly be frequent due to the monotone support property. This pruning criterion is extremely effective in reducing the search space.

```
SPADE ($min\_sup, \mathcal{D}$):
$\mathcal{F}_1$ = { frequent items };
$\mathcal{F}_2$ = { frequent 2-sequences };
for all classes $\mathcal{C}_2 \in \mathcal{F}_2$ do
    for ($k = 3; \mathcal{C}_{k-1} \neq \emptyset; k = k + 1$) do
        for all classes $[\varepsilon] \in \mathcal{C}_{k-1}$ do
            for all sequences $\alpha \in [\varepsilon]$ do
                for all sequences $\beta \in [\varepsilon], \beta \geq \alpha$ do
                    if ($|\mathcal{L}(\alpha) \cap_t \mathcal{L}(\beta)| \geq min\_sup$) then
                        $\mathcal{S} = \alpha \cup_t \beta$
                        $[\aleph_\mathcal{S}] = [\aleph_\mathcal{S}] \cup \mathcal{S}$
            $\mathcal{C}_k = \mathcal{C}_k \cup [\aleph_\mathcal{S}]$;
```

Figure 5. The SPADE Algorithm

For applying global pruning across all equivalence classes, all the cross class links have to maintained, which corresponds to storing all frequent sequences in memory. If memory is limited, then only local pruning within a class can be applied.

For fast frequency computation, SPADE maintains, for each distinct item, an inverted list (denoted $\mathcal{L}$) of *(PlanId, EventTime)* pairs where the item occurs. For example, from our initial database in Figure 3, we obtain, $\mathcal{L}(Move) = \{(1,10)(1,30)(2,10)(2,20)\}$, and $\mathcal{L}(Flat) = \{(1,30)(2,10)\}$. To compute the support of a sequence from any two of its subsets, their lists are intersected in a special way. For example to obtain $\mathcal{L}(Move, Flat) = \{(1,30)(2,10)\}$, an equality check is made for each pair, and to obtain $\mathcal{L}(Move \mapsto Flat) = \{(1,30)\}$, a check is made whether there exists any *EventTime* for *Flat* that follows any *EventTime* for *Move*, for pairs with the same *PlanId.* The reverse holds if when finding $\mathcal{L}(Flat \mapsto Move) = \{(2,20)\}$ We use the notation $\mathcal{L}(Move) \cap_t \mathcal{L}(Flat)$ for the ***temporal intersection*** refering to all three cases. When intersecting the lists for *Move* and *Flat*, SPADE simultaneously checks for the three possible candidate sequences, $Move \mapsto Flat$, $(Move, Flat)$, and $Flat \mapsto Move$. We use the notation $Move \cup_t Flat$ to denote these three possibilities. Please see [Zak01] for more details.

5. Mining Strong Sequence Rules

We now describe our methodology for extracting the predictive sequences on a sample plan database. Let $\mathcal{D}_g$ and $\mathcal{D}_b$ refer to the good and bad plans, respectively. All experiments were performed on an SGI machine with a 100MHz MIPS processor and 256MB main memory, running IRIX 6.2.

5.1. Mining the Whole Database ($\mathcal{D} = \mathcal{D}_g + \mathcal{D}_b$)

We used an example database with 522 items, 1000 good plans and 51 bad plans, with an average of 274 events per good plan, 196 events per bad plan, and an average event length of 6.3 in both. We mined the entire database of good and bad plans for frequent sequences. Since there are about 5% bad plans, we would have to use a minimum support of at least 5% to discover patterns that have some failure condition. However, even at 100% minimum support, the algorithm proved to be intractable. For example, we would find more than a 100 length sequence of the form *Move* $\cdots \mapsto$ *Move*, all 2^{100} of whose subsequences would also be frequent, since about half of the events contain a *Move*. Such long sequences would also be discovered for other common items such as *Success*, *Truck*, etc. With this high level of item frequency, and long plan-sequences, we would generate an exponential number of frequent patterns. Mining the whole database is thus infeasible. Note also that none of these rules can have high confidence, i.e., none can be used to predict plan failure, because they occur in all the good as well as the bad plans. The problem here is that the common strategy of mining for all highly frequent rules and then eliminating all the low confidences ones will be infeasible in this highly structured database.

5.2. Mining the Bad Plans ($\mathcal{D}_b$)

Since we are interested in rules that predict failure, we only need to consider patterns that are frequent in the failed plans. A rule that is frequent in the successful plans cannot have a high confidence of predicting failure. To reduce the plan-sequence length and the complexity of the problem, we decided to focus only on those events that had an outcome other than a *Success*. The rationale is that the plan solves its goal if things go the way we expect, and so it is reasonable to assume that only non-successful actions contribute to failure. We thus removed all actions with a successful outcome from the database of failed plans, obtaining a smaller database of bad plans, which had an average of about 8.5 events per plan.

The number of frequent sequences of different lengths for various levels of minimum support are plotted in Figure 6, while the running times and the total number of frequent sequences is shown in Table 1. At 60% support level we found an overwhelming number of patterns. Even at 75% support, we have too many patterns (38386), most of which are

Table 1. Discovered Patterns and Running Times

	MS=100%	MS=75%	MS=60%
#Sequences	544	38386	642597
Time	0.2s	19.8s	185.0s

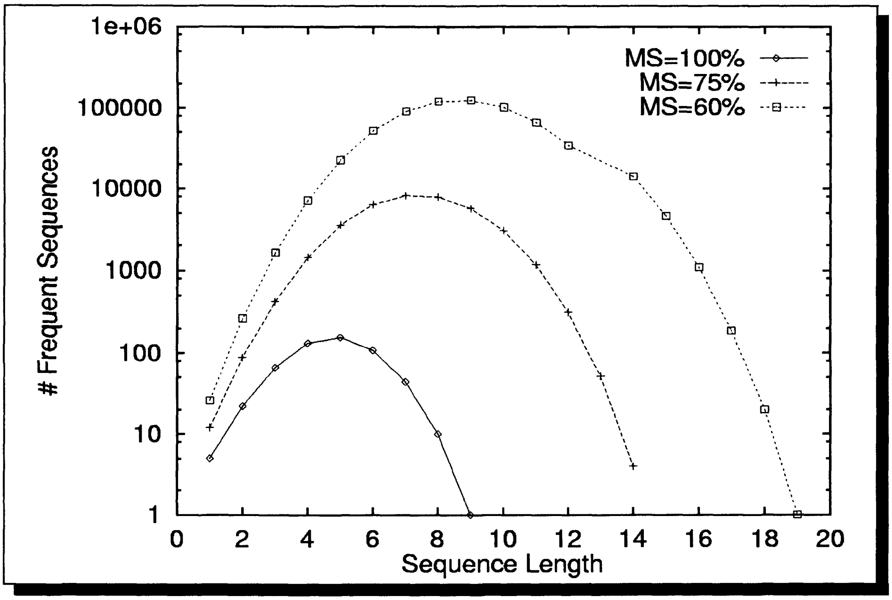


Figure 6. Sequences of Different Length Mined at Various Levels of Minimum Support (MS)

quite useless when we compute their confidence relative to the entire database of plans. For example, the pattern *Move* $\mapsto$ *Truck-1* $\mapsto$ *Move* had a 100% support in the bad plans. However, it is not at all predictive of a failure, since it occurs in every plan, both good and bad. The problem here is that if we only look at bad plans, the confidence of a rule is not an effective metric for pruning uninteresting rules. In particular, every frequent sequence α will have 100% confidence, since $fr(\alpha \mapsto Failure, \mathcal{D}_b)$ is the same as $fr(\alpha, \mathcal{D}_b)$. However, all potentially useful patterns are present in the sequences mined from the bad plans. We must, therefore, extract the interesting ones from this set. We can also reduce the number of patterns generated by putting sequence length or event width restrictions, by incorporating maximum and minimum gaps, time windows, and so on [Zak00].

5.3. Extracting Interesting Rules

A discovered pattern may be uninteresting due to various reasons [KMR+94]. For example, it may correspond to background knowledge, or it may be redundant, i.e., subsumed by another equally predictive but more general pattern. Below we present our pruning schemes for retaining only the most predictive patterns.

5.3.1 Pruning Normative Patterns.

Background knowledge plays an important role in data mining [FPSSU96]. One type of background knowledge, which we call *normative* knowledge, corresponds to a set of patterns that are uninteresting to the user, often because they are obvious. Normative knowledge can be used to constrain or prune the search space, and thereby enhance the performance. Typically, the normative knowledge is hand-coded by an expert who knows the domain. In our case normative knowledge is present in the database $\mathcal{D}_g$ of good plans. The good plans describe the normal operations, including the minor problems that may arise frequently but do not lead to plan failure. We automatically extract the normative knowledge from the database of good plans as follows: We first mine the bad plans $\mathcal{D}_b$ for frequent sequences. We also compute the support of the discovered sequences in the successful plans. We then eliminate those sequences that have a high support (greater than a user-specified *max_sup* in $\mathcal{D}_g$) in the successful plans, since such sequences represent the normal events of successful plans. This automatic technique for incorporating background knowledge is effective in pruning the uninteresting patterns. Figure 7 shows the reduction in the number of frequent sequences by excluding normative patterns. At 25% maximum support in $\mathcal{D}_g$, we get more than a factor of 2 reduction (from 38386 to 17492 rules).

5.3.2 Pruning Redundant Patterns.

Even after pruning based on normative knowledge, we are left with many patterns (17492), which have high frequency and high confidence, i.e., they are highly predictive of failure. The problem is that the existence of one good rule implies the existence of many almost identical, and equally predictive rules. For example, suppose *(Flat Truck-1)* $\mapsto$ *(Overheat Truck-1)* is highly predictive, and that the first action of every plan is a *Move*. In this case *Move* $\mapsto$ *(Flat Truck-1)* $\mapsto$ *(Overheat Truck-1)*, will be equally predictive, and will have the same frequency. The latter sequence is thus redundant. Formally, β is *redundant* if there exists $\alpha \prec_1 \beta$, with the same support as β both in good and bad plans (recall that $\alpha \prec_1 \beta$, if α is obtained by removing a single item from β).

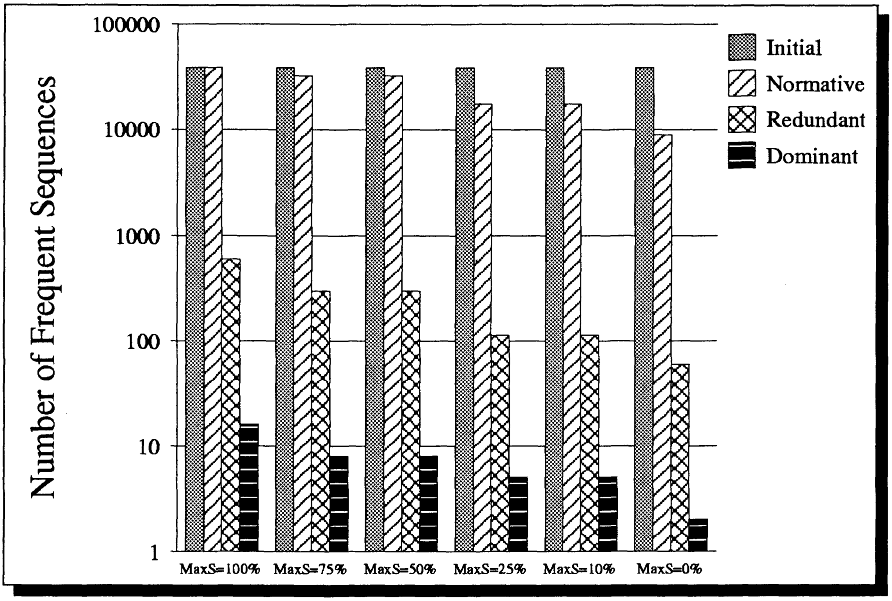


Figure 7. Effect of Different Pruning Techniques

Given the high frequency of some actions in our domain, there is tremendous redundancy in the set of highly predictive and frequent patterns obtained after normative pruning. Therefore, we prune all redundant patterns. Figure 7 shows that by applying redundant pruning in addition to normative pruning we are able to reduce the pattern set from 17492 down to 113. This technique is thus very effective.

```
/* Mine Bad Plans */
1. $\mathcal{I}$ = SPADE $(min\_sup, \mathcal{D}_b)$
/* Prune Normative Patterns */
2. $\mathcal{H} = \{\alpha \in \mathcal{I} \mid fr(\alpha, \mathcal{D}_g) < max\_sup\}$
/* Prune Redundant Patterns */
3. $\mathcal{R} = \{\alpha \in \mathcal{H} \mid \nexists \beta \succ_1 \alpha$ such that $fr(\alpha, \mathcal{D}_b) = fr(\beta, \mathcal{D}_b)$
        and $fr(\alpha, \mathcal{D}_g) = fr(\beta, \mathcal{D}_g)\}$
/* Prune Dominated Patterns */
4. $\mathcal{F} = \{\alpha \in \mathcal{R} \mid \nexists \beta \succ_1 \alpha$ such that $fr(\alpha, \mathcal{D}_b) \geq fr(\beta, \mathcal{D}_b)$
        and $fr(\alpha, \mathcal{D}_g) \leq fr(\beta, \mathcal{D}_g)\}$
```

Figure 8. The Complete PLANMINE Algorithm

5.3.3 Pruning Dominated Patterns. After applying normative and redundant pruning, there still remain some patterns that are very similar. Above, we pruned rules which had equivalent support. We can also prune rules based on confidence. We say that β is *dominated* by α, if $\alpha \prec_1 \beta$, and α has lower support in good and higher support in bad plans (i.e., α has higher confidence than β). Figure 7 shows that dominant pruning, when applied along with normative and redundant pruning, reduces the rule set from 113 down to only 5 highly predictive patterns. The combined effect of the three pruning techniques is to retain only the patterns that have the highest confidence of predicting a failure, where confidence is given as:

$$Conf(\alpha) = \frac{fr(\alpha \mapsto Failure, \mathcal{D})}{fr(\alpha, \mathcal{D})} = \frac{|\alpha \preceq S_b \in \mathcal{D}_b|}{|\alpha \preceq S \in \mathcal{D}|} \tag{27.1}$$

Figure 8 shows the complete pruning algorithm. An important feature of our approach is that all steps are automatic. The lattice structure on sequences makes the redundancy and dominance easy to compute. Given the databases $\mathcal{D}_b$ and $\mathcal{D}_g$, *min_sup*, and *max_sup*, the algorithm returns the set of the most predictive patterns.

6. Experimental Evaluation

In this section we present an experimental evaluation of PLANMINE. We show how it is used in the TRIPS [FJ98] and IMPROVE [LMA98] applications, and how it is be used to build plan monitors.

6.1. TRIPS and IMPROVE Applications

TRIPS is an integrated system in which a person collaborates with a computer to develop a high quality plan to evacuate people from a small island. During the process of building the plan, the system simulates the plan repeatedly based on a probabilistic model of the domain, including predicted weather patterns and their effect on vehicle performance. The system returns an estimate of the plan's success. Additionally, TRIPS invokes PLANMINE on the execution traces produced by simulation, in order to analyze *why* the plan failed when it did. This information can be used to improve the plan.

It is difficult to quantify the performance of TRIPS or how much the PLANMINE component contributes to it. However, both seem to work well on our test cases. In one example, we use TRIPS to develop a plan that involves using two trucks to bring the people to the far side of a collapsed bridge near the destination city. A helicopter then shuttles the people, one at a time, to the destination city. The plan works well unless the truck with the longer route gets two or more flat tires, which delay the truck. If the truck is late, then the helicopter is also more likely to crash, since the weather worsens as time progresses. On this example, PLANMINE successfully determined that *(Move Truck1 Flat)* → *(Move Truck1 Flat)* ⇒ *Failure*, as well as *(Move Heli1 Crash)* ⇒ *Failure*, is a high confidence rule for predicting plan failure.

Table 2. Performance of IMPROVE (averaged over 70 trials).

	initial plan length	final plan length	initial success rate	final success rate	num. plans tested
IMPROVE	272.3	278.9	0.82	0.98	11.7
RANDOM	272.3	287.4	0.82	0.85	23.4
HIGH	272.6	287.0	0.82	0.83	23.0

PLANMINE has also been integrated into an algorithm for *automatically* modifying a given plan so that it has a higher probability of achieving its goal. IMPROVE [LMA98] runs PLANMINE on the execution traces of the given plan to pinpoint defects in the plan that most often lead to plan failure. It then applies qualitative reasoning and plan adaptation algorithms to modify the plan to correct the defects detected by PLANMINE. It does this by adding actions to make the patterns that predict failure less likely to occur. For example, if PLANMINE produces the rule *(Truck1 Flat)* → *(Truck1 Overheat)* ⇒ *Failure* then IMPROVE will conclude that *either* preventing *Truck*1 from getting a flat or from overheating might improve the plan. In each iteration, IMPROVE constructs several plans which might be better than the original plan. If any of the plans performs better in simulation than the original plan, then IMPROVE repeats the entire process on the new best plan in simulation. This process is repeated until no suggested modification improves the plan.

Table 2 shows the performance of the IMPROVE algorithm, as reported in [LMA98], on a large evacuation domain that contains 35 cities, 45 roads, and 100 people. The people are scattered randomly in each trial, and the goal is always to bring all the people, using two trucks and a helicopter, to one central location. The trucks can hold 25 people and the helicopter only one person, so the plans involve multiple round trips. The plans succeed unless a truck breaks down or the helicopter crashes. For each trial we generate a random set of road conditions and rules which give rise to a variety of malfunctions in the vehicles, such as a truck getting a flat tire or overheating. Some malfunctions worsen the condition of the truck and make other problems, such as the truck breaking down more likely. The process is not completely random in that by design there usually exists some sequence of two to three malfunctions that makes a breakdown or crash very likely. Furthermore, there are always several malfunctions, such as trucks getting dents or having their windows cracked, that occur frequently and never cause other problems. We use a domain-specific greedy scheduling algorithm to generate initial plans for this domain. The initial plans contain over 250 steps.

We compared IMPROVE with two less sophisticated alternatives. The RANDOM approach modifies the plan randomly five times in each iteration, and chooses the modification that works best in simulation. The HIGH approach replaces the PLANMINE component of IMPROVE with a technique that simply tries to prevent the malfunctions that occur most often. As shown in Table 2, IMPROVE with PLANMINE increases a plan's probability of achieving its goal, on average, by about 15%, but without PLANMINE only by, on average, about 3%.

6.2. Plan Monitoring

We now describe experiments to directly test PLANMINE. In each trial, we generate a training and a test set of plan executions. We run PLANMINE on the training set and then evaluate the discovered rules on the test set. We used the sequence rules to build monitors, which observe the execution of the plan, and sound an alarm when a plan failure is likely. The hypothesis behind these tests is that predicting failure accurately will be useful in avoiding errors during plan execution. We used the same evacuation domain described above. The training set had 1000 plan traces, with around 5% plan-failure rate. Only 300 of the good plans were used for background knowledge. We used a *min_sup* of 60% in the bad plans, and a *max_sup* of 20% in the good plans.

We run PLANMINE on the training data and use the discovered set of rules $\mathcal{R}$ to build a *monitor* – a function that takes as input the actions executed so far and outputs failure *iff* any of the rules in $\mathcal{R}$ is a subsequence of the action sequence. For example, a monitor built on the rules *(Truck-1 Flat)* $\mapsto$ *(Truck-1 Overheat)* $\Rightarrow$ *Failure* and *(Truck-2 Flat)* $\mapsto$ *(Truck-2 Flat)* $\Rightarrow$ *Failure* sounds its alarm if *Truck-1* gets a flat tire and overheats, or if *Truck-2* gets two flat tires. The *precision* of a monitor is the percentage of times the monitor signals a failure, and a failure actually occurs (i.e., the ratio of correct failure signals to the total number of failure signals). The *recall* of a monitor is the percentage of failures signaled prior to their occurrence. A monitor that always signals failure has 100% recall and p% precision where p is the rate of plan failure. To generate monitors, first we mine the database of execution traces for sequence rules. We then build a monitor by picking some threshold λ, varied in the experiments, and retain only those rules that have at least λ precision or confidence (see Equation 27.1) on the training data.

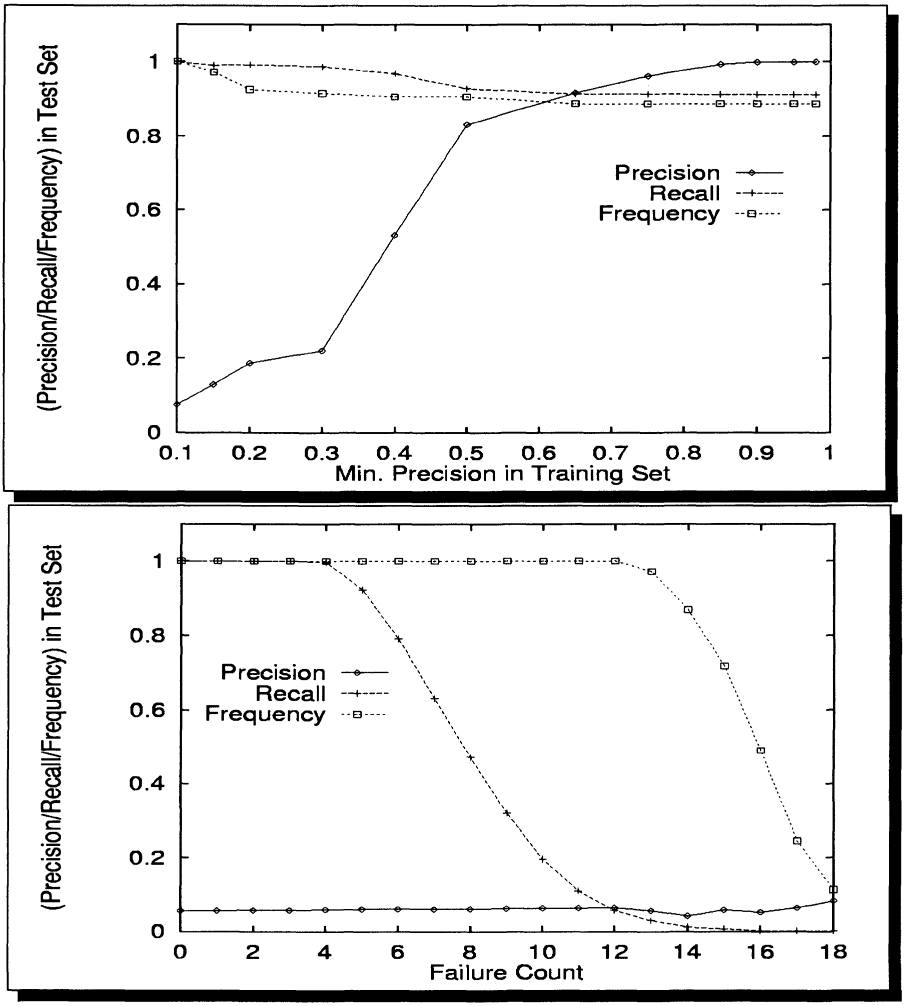


Figure 9. a) Using PLANMINE for Prediction; b) Using Failure Count for Prediction

Figure 9a shows the evaluation of the monitors produced with PLANMINE on a test set of 500 (novel) plans. The results are the averages over 105 trials, and thus each number reflects an average of approximately 50,000 separate tests. It plots the precision, recall, and frequency of the mined rules in the test database against the precision threshold in the training data. The frequency graph shows the percent of trials for which we found at least one rule of the given precision. The figure clearly shows that our mining and pruning techniques produce excellent monitors, which have 100% precision with recall greater than 90%. We can produce monitors with significantly higher recall, but only by reducing precision to around 50%. The desired tradeoff depends on the application. If plan failures are very costly then it might be worth sacrificing precision for recall. For comparison we also built monitors that signaled failure as soon as a fixed number of malfunctions of any kind occurred. Figure 9b shows that this approach produces poor monitors, since there was no correlation between the number of malfunctions and the chance of failure (precision).

We also investigated whether or not data mining was really necessary to obtain these results. The graphs in Figure 10 describe the performance of the system if we limit the length of the rules. For example, limiting the rules to length two corresponds to building a monitor out of the pairs of actions that best predict failure. The figure shows the precision, recall, and frequency of the rules of different lengths discovered in the test database plotted against the precision threshold in the training data. The frequency graph shows the percent of trials for which we found at least one rule of the given precision and the given length. For example, at 0.5 training precision, out of the 105 trials, we found a frequent rule of length 3 in more than 90% of the trials, and a rule of length 5 in 10% of the trials, and so on.

The results indicate that monitors built out of rules of length less than three are much worse than monitors built out of longer rules. In particular, the graphs show that there were very few rules of length one or two with even 50% or higher precision. Furthermore, rules of higher length always had better recall for the same level of precision. However, only 30% of our experiments produced useful rules of length four and only 10% produced rules of length five. But when these rules were produced, they were highly effective.

7. Conclusions

We presented PLANMINE, an automatic mining method that discovers event sequences causing failures in plans. We developed novel pruning techniques to extract the set of the most predictive rules from highly structured plan databases. Our pruning strategies reduced the size of the rule set by three orders of magnitude. The rules discovered by PLANMINE were extremely useful for understanding and improving plans, as well as for building monitors that raise alarms before failures happen, i.e., we show that not only can we analyze the simulation traces of a sin-

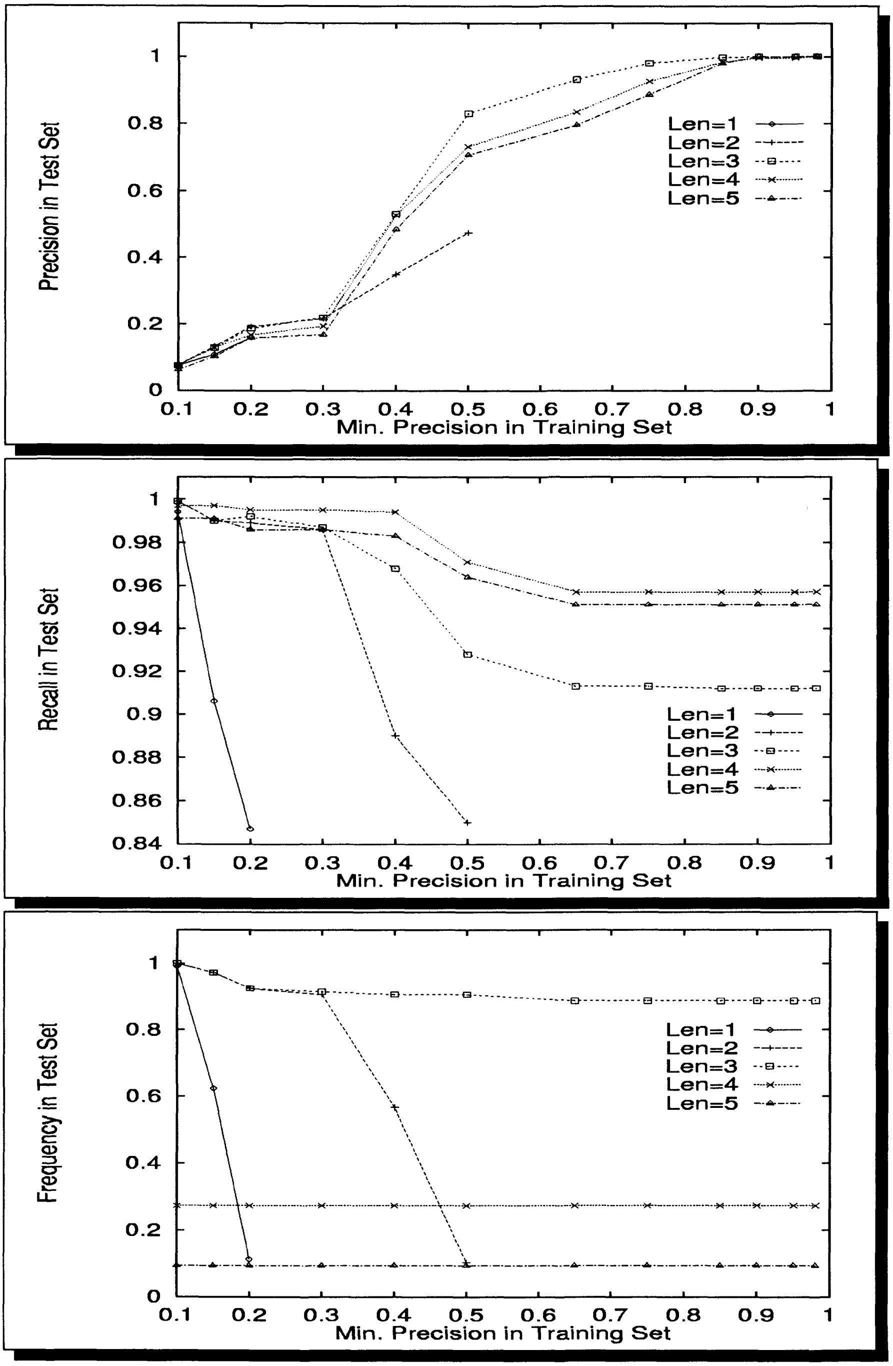


Figure 10. a) Precision, b) Recall and c) Frequency of Discovered Sequences in Test Set

gle plan to improve it, but we also can analyze multiple plans executions and detect common failure modes.

There are several directions in which this work can be extended. In the experiments we were limited to using a 60% minimum support. A lower value would easily generate more than a million patterns. But a high support value can miss long failure sequences. It will only find fragments of the long sequences. We would thus like to be able to lower the minimum support threshold, and we would like to do this without making the problem intractable. One reason for the combinatorial explosion of patterns with decreasing support is that we do not impose any restrictions on the event times. It might be reasonable to assume that failures are caused by events that follow closely in time, for example only within a specific window of time. Such constraints can significantly reduce the number of patterns, and can enable us to mine at lower support levels. Other types of constraints include putting restrictions on the sequence lengths, minimum and maximum gaps between successive sequence elements, and so on.

In our current approach we first mine the bad plans, and then apply the pruning in a separate step by comparing the support in both the good and bad plans. A promising direction is to incorporate the pruning directly into the first step, and to mine both the databases simultaneously. This can result in significant speedups by pruning patterns as early in the computation as possible. One can perhaps use information about the planner and the kinds of action sequences that can even be generated to improve the efficiency of this application significantly.

Using support as a percentage of the whole plan database can also be potentially limiting. For example, if the planner performs a wide variety of plans for differing goals, we would need a lower support threshold to compensate for the diversity. While incorporating constraints on discovered patterns is one solution, an alternative would be to change the denominator in the frequency formula to reflect similarity in the goals of the plans. It would also be interesting to study the long term effects of data mining, i.e., what happens if the process is repeated on new traces? How does one merge new rules with the existing ones? and so on.

References

[AMS97] K. Ali, S. Manganaris, and R. Srikant. Partial Classification using Association Rules. In *3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining*, August 1997.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *11th Intl. Conf. on Data Engg.*, 1995.

[Bay97] R. J. Bayardo. Brute-force mining of high-confidence classification rules. In *3rd Intl. Conf. on Knowledge Discovery and Data Mining*, August 1997.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, 1984.

[FJ98] G. Ferguson and A. James. TRIPS: An Integrated Intelligent Problem-Solving Assistant . In *15th Nat. Conf. AI*, 1998.

[FPSSU96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, 1996.

[Ham90] K. Hammond. Explaining and repairing plans that fail. *J. Artificial Intelligence*, 45:173–228, 1990.

[HC95] A. E. Howe and P. R. Cohen. Understanding planner behavior. *J. Artificial Intelligence*, 76(1):125–166, 1995.

[HKM$^+$96] K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. Knowledge discovery from telecommunication network alarm databases. In *12th Intl. Conf. Data Engineering*, February 1996.

[HV98] K. Z. Haigh and M. M. Veloso. Learning situation-dependent costs: Improving planning from probabilistic robot execution. In *Intl. Conf. Autonomous Agents*, May 1998.

[KHW95] N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. *J. Artificial Intelligence*, 76:239–286, 1995.

[KMR$^+$94] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *3rd Intl. Conf. Information and Knowledge Management*, pages 401–407, November 1994.

[LMA98] N. Lesh, N. Martin, and J. Allen. Improving big plans. In *15th Nat. Conf. AI*, 1998.

[McD94] D. McDermott. Improving robot plans during execution. In *2nd Intl. Conf. AI Planning Systems*, pages 7–12, June 1994.

[Min90] Steven Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42(2–3), 1990.

[MTV97] H. Mannila, H. Toivonen, and I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and*

Knowledge Discovery: An International Journal, 1(3):259–289, 1997.

[OC96] T. Oates and P. R. Cohen. Searching for planning operators with context-dependent and probabilistic effects. In *13th Nat. Conf. AI*, 1996.

[RSE94] P. Riddle, R. Segal, and O. Etzioni. Representation design and brute-force induction in a boeing manufacturing domain. *Applied Aritficial Intelligence*, 8:125–147, 1994.

[RT94] G. Reece and A. Tate. Syntehsizing portection monitors from causal structure. In *2nd Intl. Conf. AI Planning Systems*, June 1994.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *5th Intl. Conf. Extending Database Technology*, March 1996.

[Zak98] M. J. Zaki. Efficient enumeration of frequent sequences. In *7th Intl. Conf. on Information and Knowledge Management*, November 1998.

[Zak00] M. J. Zaki. Sequences mining in categorical domains: Incorporating constraints. In *9th Intl. Conf. on Information and Knowledge Management*, November 2000.

[Zak01] M. J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1/2), Jan/Feb 2001. Special issue on Unsupervised Learning.

Chapter 28

EFFICIENT ALGORITHMS FOR MINING LONG PATTERNS IN SCIENTIFIC DATA SETS

Ramesh C. Agarwal and Charu C. Aggarwal

Abstract In this paper we present an algorithm for mining long patterns in databases. The algorithm finds large itemsets by using depth first search on a lexicographic tree of itemsets. The focus of this paper is to develop CPU-efficient algorithms for finding frequent itemsets in the cases when the database contains patterns which are very wide. We refer to this algorithm as *DepthProject*, and it achieves upto two orders of magnitude speedup over the recently proposed *MaxMiner* algorithm for finding long patterns. These techniques may be quite useful for applications in areas such as computational biology in which the number of records is relatively small, but the itemsets are very long. This necessitates the discovery of patterns using algorithms which are especially tailored to the nature of such domains.

Keywords: Long Patterns, Association Rules, Itemsets, Scientific Data Sets

1. Introduction

The association rule problem has been recognized in the literature as a fundamentally important problem in the field of data mining. Applications of association rules extend to finding useful patterns in consumer behavior, target marketing, computational biology, and electronic commerce. The association rule model was introduced by Agrawal, Imielinski, and Swami [AIS93].

Let I be a set of items. Each transaction T in the database is a subset of the items occuring in I. An *association rule* is a condition of the form $X \Rightarrow Y$ where $X \subseteq I$ and $Y \subseteq I$ are two sets of items. The intuitive meaning of an association rule is that the presence of the set of items X

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 541–566.
© 2001 *Kluwer Academic Publishers.*

in a transaction is a strong indicator of the possibility of the presence of the set of items Y.

A set of items in the database is referred to as an *itemset.* In particular, a set of items with a cardinality of k is referred to as a k-itemset. The measures used to evaluate the strength of an association rule are *support* and *confidence.* The support of the rule $X \Rightarrow Y$ is the fraction of customers who buy both X and Y. The confidence of the rule $X \Rightarrow Y$ is the fraction of customers buying X who also buy Y.

Starting with the pioneering work in [AIS93], the association rule problem and its variations have been studied extensively by researchers. In addition, a number of methods have been discussed in the literature which extend the binary association rule problem to related scenarios such as quantitative association rules, and generalized association rules [SA95, SA96]. Methods for providing ad-hoc query capabilities and integrating association rule mining with database systems have been discussed in [LNHP99].

In the initial work proposed by Agrawal et. al.[AIS93], a two-phase method was proposed. In the first phase, all large itemsets were generated, whose support is above the user-specified minimum support. In the second phase, these large itemsets are used in order to generate association rules whose confidence is above the user-specified minimum confidence. The bottleneck operation in this algorithm is the first phase in which large itemsets are generated. A faster algorithm called *Apriori* was proposed in [AMS94], which generates $(k+1)$-candidates using joins over frequent k-itemsets which were already generated. Thus, for each frequent itemset, all subsets of it need to be generated by the algorithm.

This paper discusses the problem of mining long patterns in scientific data sets. Such data sets typically have two properties:

- The length of a pattern may be very high. For example, in a computational biology application, a frequently occuring pattern may have length of upto 50 or 100.
- The number of transactions in the data is much smaller than a market basket application. Market basket applications are typically based on very large scale consumer behavior. This results in huge data sets which often grow without limit. On the other hand, scientific applications such as computational biology tend to be far more conservative in the number of transactional patterns that can be stored. Thus, this results in databases which are smaller than the typical market basket applications.

Because of the inherent difficulty of the itemset generation problem in terms of computational complexity, considerable research has been devoted towards finding faster methods for generating large itemsets [AMS94, BAY98, DS99, GMS97, HPY00, HTV94, SON95, TO96, ZPOL97]. The large itemset problem is reasonably well solved at least for the case of very sparse sales transaction data, when the pattern lengths are short [AAV01, AMS94]. An interesting analysis of the impact of different kinds of data on access costs has been provided in [DS99]. An *Apriori*-style algorithm with improved counting techniques using columnwise data access for databases with a larger number of items has also been discussed in the same work. We maintain that when the actual frequent patterns are wide, even the CPU-costs of any algorithm which is based on the *Apriori*-framework would be compromised by the investigation of all 2^k subsets of frequent k-patterns. In such cases, the frequent itemset generation algorithms become CPU-bound. Some of the algorithms in the literature such as *MaxMiner* avoid this by implementing *lookaheads* [BAY98], in which supersets of frequent patterns are used in order to prune off potential candidates in the search. In spite of these advances [BAY98, DS99], finding computationally efficient algorithms for generating long patterns continues to be a very difficult problem. This paper[1] is written with the primary aim of developing an algorithm for long patterns which is CPU-efficient.

This paper is organized as follows. In section 2, we will discuss several formalizations, which will be useful in describing the algorithm. In section 3, we will introduce the depth-first strategy of generating large itemsets. The database representation will be explained in section 4. In section 5, we will discuss the bucketing technique which is a useful method for speeding up the algorithm substantially. An effective pruning and lookahead strategy for the algorithm is discussed in section 6. We will compare the results to the *MaxMiner* algorithm, and present the computational results in section 7. Section 8 discusses the conclusions and summary.

1.1. Contributions and Motivation

This paper proposes a fundamentally different technique for finding large itemsets from most of the algorithms proposed in the recent past. A large number of algorithms in the research literature start the process of generating $(k+1)$ itemsets only after all k-itemsets have been generated. Even the recent look-ahead-based algorithm discussed [BAY98] for mining long patterns guarantees the discovery of all $(k+1)$-itemsets only after generation of all k-itemsets, even though some of the $(k+1)$-itemsets may be discovered earlier than the k-itemsets. The reason for this natural algorithmic design has been motivated by the desire to restrict the number of passes over the database to the length of the longest pattern. This often results in the generation of a large number of subsets of frequent itemsets. A few methods [BMUT97] deviate from this natural design in order to reduce the number of I/O passes, but tend to be *Apriori*-like in their overall approach; consequently the combinatorial explosion problem continues to be an issue.

The long pattern problem is so difficult to solve computationally, that even for databases of relatively small sizes, it may be very difficult to find long patterns [BAY98]. In the past decade, memory availability has increased by orders of magnitude. It has recently starting becoming increasingly evident that in the near future, many medium to large size databases are likely to be main memory resident. For problems in which the patterns are longer than 15-20 items, and the database is too large to fit under the current memory limitations (which are reaching the Gigabyte order), most of the algorithms which require the generation of all subsets of frequent itemsets are impractical anyway. For the *long pattern* problem, it may perhaps be realistic to design algorithms with much greater focus on CPU requirements for transaction sets of moderate sizes which can fit in main memory.[2]

The long pattern problem has considerable applicability in combinatorial pattern discovery in biological sequences [RF98]. In this case, we have a small set of biological sequences, and we wish to find all the rigid patterns (motifs) in these sequences. This is an example of a problem in which the number of patterns is small enough to fit in main memory, whereas each individual pattern can be quite long, and it may be too time consuming to use *Apriori*-like algorithms for such problems. Our results in this paper are tailored towards such domains in which the patterns are too long to be handled by *Apriori*-style algorithms.

2. The Lexicographic Tree of Itemsets

In this section, we will introduce a conceptual representation of large itemsets which we will refer to as the lexicographic tree [AAV01]. This will provide us with the necessary machinery needed in order to explain the *DepthProject* algorithm. We assume that a lexicographic ordering exists among the items in the database. In order to indicate that an item i occurs lexicographically earlier than j, we will use the notation $i \leq_L j$. The lexicographic tree is an abstract representation of the large itemsets with respect to this ordering. The lexicographic tree is defined in the following way:

(1) A node exists in the tree corresponding to each large itemset. The root of the tree corresponds to the null itemset.

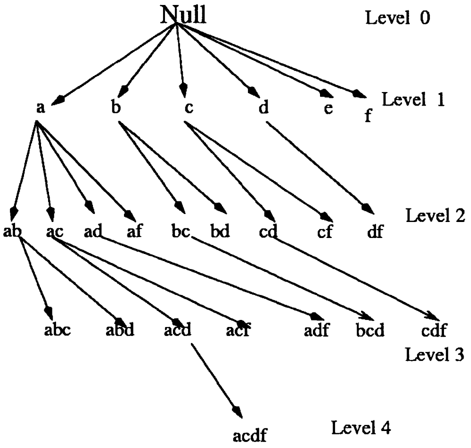


Figure 1. The lexicographic tree

(2) Let $I = \{i_1, \dots i_k\}$ be a large itemset, where $i_1, i_2 \dots i_k$ are listed in lexicographic order. The parent of the node I is the itemset $\{i_1, \dots i_{k-1}\}$.

This definition of ancestral relationship naturally defines a tree structure on the nodes, which is rooted at the *null* node. The goal in this paper is to use the structure of the lexicographic tree in order to substantially reduce the CPU time for counting large itemsets. An example of the lexicographic tree is illustrated in Figure 1. A frequent 1-extension of an itemset such that the last item is the contributor to the extension will be called a *frequent lexicographic tree extension*, or simply a tree extension. Thus, each edge in the lexicographic tree corresponds to an item which is the frequent lexicographic tree extension to a node. We will denote the set of frequent lexicographic tree extensions of a node P by $E(P)$. In the example illustrated in Figure 1, the frequent lexicographic extensions of node a are b, c, d, and f.

Let Q be the immediate ancestor of the itemset P in the lexicographic tree. The set of *prospective branches* of a node P is defined to be those items in $E(Q)$ which occur lexicographically after the node P. These are the *possible* frequent lexicographic extensions of P. We denote this set by $F(P)$. Thus, we have the following relationship: $E(P) \subseteq F(P) \subset E(Q)$. The value of $E(P)$ in Figure 1, when $P = ab$ is $\{c, d\}$. The value of $F(P)$ for $P = ab$ is $\{c, d, f\}$, and for $P = af$, $F(P)$ is empty.

A node is said to be *generated*, the first time its existence is discovered by virtue of the extension of its immediate parent. A node is said to have been *examined*, when its frequent lexicographic tree extensions have been determined. Thus, the process of examination of a node P results in generation of further nodes, unless the set $E(P)$ for that node is empty. Obviously a node can be examined only after it has been generated. This paper will discuss an algorithm which constructs the lexicographic tree in depth-first order by starting at the node *null* and successively generating nodes until all nodes have been generated and subsequently examined.

An itemset is defined to be a *frequent maximal itemset*, if it is frequent, and no superset of that itemset is frequent. Thus, in Figure 1, the itemset *acdf* is maximal.

Let P be a node in the lexicographic tree corresponding to a frequent k-itemset. Then, for a transaction T we define the *projected transaction* $T(P)$ to be equal to $T \cap E(P)$. However, if T does not contain the itemset corresponding to node P then $T(P)$ is null. For a set of transactions $\mathcal{T}$, we define the projected transaction set $\mathcal{T}(P)$ to be the set of projected transactions in $\mathcal{T}$ with respect to frequent items $E(P)$ at P.

Consider the transaction $abcdefghk$. Then, for the example A of Figure 1, the projected transaction at node *null* would be $\{a, b, c, d, e, f, g, h, k\} \cap \{a, b, c, d, e, f\} = abcdef$. The projected transaction at node a would be $bcdf$. For the transaction $abdefg$, its projection on node ac is null because it does not contain the required itemset ac.

We emphasize the following points:

(1) For a given transaction T, the information required to count the support of any itemset which is a descendant of a node P is completely contained in $T(P)$.

(2) The number of items in a projected transaction $T(P)$ is typically much smaller than the original transaction.

(3) For a given transaction set $\mathcal{T}$ and node P the ratio of the number of transactions in $\mathcal{T}(P)$ and $\mathcal{T}$ is approximately determined by the support of P.

In the next section, we will discuss an algorithm which finds the itemsets by depth first creation of the lexicographic tree. The following description will discuss a pure approach in which the entire tree is generated by the algorithm. In a later section, we will show that it is possible to significantly speed up the algorithm by pruning away those subtrees which are guaranteed to contain only non-maximal itemsets. In addition, the discussion in the following section does not include a

```
Algorithm DepthFirst(Itemset Node: $N$, PointerToDatabase: $\mathcal{T}$, Bitvector : $B$)
  begin
  $C = GenerateCandidates(N)$
  $E = Count(N, \mathcal{T}, B, C)$
  { Let $E = \{i_1, \ldots, i_{|E|}\}$, when expressed in lexicographic order }
  write each of the frequent itemsets $N \cup \{i_r\}$ for $r \in \{1, \ldots, |E|\}$ to disk;
  $B' = CreateBitvector(N, B, \mathcal{T})$;
  if $(ProjectionCondition)$ then
    begin
    $\mathcal{T}' = Project(\mathcal{T}, E, N, B')$;
    Modify $B'$ to be a set of ones, one for each element in $\mathcal{T}'$;
    end;
       else $\mathcal{T}' = \mathcal{T}$;
  for $r := 1$ to $|E|$
     begin
     $DepthFirst(N \cup \{i_r\}, \mathcal{T}', B')$
     end
  end

Subroutine Project(Database: $\mathcal{T}$, FrequentExtensions : $E$, Bitvector: $B$)
  begin
  $\mathcal{T}'$ = Empty set of transactions;
  for each transaction $T \in \mathcal{T}$ do
    begin
    if corresponding bit in $B$ is 1 then add $T \cap E$ to $\mathcal{T}'$
    end
  return($\mathcal{T}'$)
  end

Subroutine CreateBitvector($N, B, \mathcal{T}$)
  begin
  Initialize $B' = B$;
  Let $n$ be the lexicographically largest item in $N$;
  for each transaction $T \in \mathcal{T}$ do
     if $n \notin T$ then set the corresponding bit in $B'$ to 0;
  return($B'$);
end
```

Figure 2. Finding the projected transaction sets at the various nodes

specialized counting technique (called bucketing) for lower level nodes, which we will also postpone to a later section.

3. Depth First Creation of the Lexicographic Tree

In depth-first search, the nodes of the lexicographic tree are *examined* in depth-first order. The process of examination of a node refers to the counting of the supports of the candidate extensions of the node. In other words, the support of all descendant itemsets of a node is determined before determining the frequent extensions of other nodes of

```
Algorithm GenerateCandidates(Itemset Node:N)
begin
  if (N is null) then
      return all items;
  else
      return all frequent extensions of parent of N which
          are lexicographically larger than any item in N;
end
```

Figure 3. Generating candidate branches of a node

```
Algorithm Count(Itemset Node:N, Pointer to Database: $\mathcal{T}$, Bitvector: B, Candidate Set: C)
begin
   Count all the possible frequent extensions of node N
   denoted by candidate set C using the database $\mathcal{T}$ and
   the bitvector B. Details of the counting procedure are provided
   in Section 4.1;
end
```

Figure 4. Counting frequent extensions of a node

the lexicographic tree. Thus, the order in which a depth-first search method would count the extensions of nodes in the Figure 1 is *null*, *a*, *ab*, *abc*, *abd*, *ac*, *acd*, *acdf*, *acf*, *ad*, *adf*, *af*, *bc*, *bcd*, *bd*, *c*, *cd*, *cdf*, *cf*, *d*, *df*, *e*, and *f*. Thus, the depth first strategy quickly tends to find the longer patterns first in the search process. At any point in the search, we maintain the projected transaction sets for some of the nodes on the path from the root to the node which is currently being extended.[3] A pointer is maintained at each node P to the projected transaction set which is available at the nearest ancestor of P. Since the projected database is substantially smaller than the original database both in terms of the number of transactions, and the number of items, the process of finding the support counts is quickened up substantially. The following information is stored at each node during the process of construction of the lexicographic tree:

(1) The itemset P at that node.

(2) The set of lexicographic tree extensions at that node which are $E(P)$.

(3) A pointer to the projected transaction set $\mathcal{T}(Q)$, where Q is some ancestor of P (including itself). The root of the tree points to the entire transaction database.

(4) A bitvector containing the information about which transactions contain the itemset for node P as a subset. The number of bits in

this bitvector is equal to the total number of transactions in $\mathcal{T}(Q)$. The value of a bit for a transaction is equal to 1, if the itemset P is a subset of the transaction. Otherwise it is equal to zero. Thus, the number of 1 bits is equal to the number of transactions in $\mathcal{T}(Q)$ which project to P. The bitvectors are used in order to make the process of support counting more efficient.

Once we have identified all the projected transactions at a given node, then finding the subtree rooted at that node is a completely independent itemset generation problem with a *substantially reduced* transaction set. As was indicated earlier, the number of transactions at a node is proportional to the support at that node. An important fact about projections is the following:

By using hierarchical projections, we are reusing the information from counting k-itemsets in order to count $(k+1)$-itemsets.

Note that such a reuse of information is made possible by the depth first strategy, since we only need to maintain the projected transaction sets on the path of the tree which is currently being explored. Let us consider a k-itemset I at which the database is projected. If a transaction T does not contain this k-itemset I as a subset, then the projection strategy ensures that T will not be used in order to count any of any of the $(k+1)$-extensions of I. This is important in reducing the running time, since a large fraction of the transactions will not be relevant in counting the support of an itemset. Furthermore, the process of projection reduces the number of fields in the database to a small number so that the counting process becomes more efficient.

The description in Figure 2 shows how the depth first creation of the lexicographic tree is performed. The algorithm is described recursively, so that the call from each node is a completely independent itemset generation problem, which finds all frequent itemsets that are descendants of a node. There are three parameters to the algorithm, a pointer to the database $\mathcal{T}$, the itemset node N, and the bitvector B. The bit vector B contains one bit for each transaction in $T \in \mathcal{T}$, and indicates whether or not the transaction T should be used in finding the frequent extensions of N. A bit for a transaction T is one, if the itemset at that node is a subset of the corresponding transaction. The first call to the algorithm is from the *null* node, the parameter $\mathcal{T}$ is the entire transaction database. Since each transaction in the database is relevant in order to perform the counting, the bitvector B consists of all "one " values.

The first step of the algorithm is to generate all the candidate extensions of N. This is accomplished by the call *GenerateCandidates*(N). For the case of the *null* node, this call returns all the items in the entire database. For the case of other nodes, the procedure simply returns the set of items in $F(N)$. As discussed earlier, the set of items in $F(N)$ may be determined by finding all those frequent extensions of the parent of N, which are lexicographically larger than any item in N. The details of the *GenerateCandidates*$(\cdot)$ procedure are illustrated in Figure 3. The next step is to count the support of each of these candidate extensions. This is done by the procedure *Count*$(\cdot,\cdot,\cdot,\cdot)$. The bitvector B and the database $\mathcal{T}$ are used in order to perform the counting efficiently, since it uses information from counting ancestral nodes. The details of how the counting is performed will be discussed in a later section. At the same time, it is desirable to create the bitvectors for node N. The bit vectors for node N may be derived very simply from the bitvector B. This is because the bitvector B corresponds to the parent of node N, which differs from N by exactly one item (the lexicographically largest item). Let n be the lexicographically largest item in N. Thus, the new bitvector for node N may be obtained by changing those bits in B to 0, such that the corresponding transactions do not contain the item n.

For some of the nodes, it is desirable to project the database, if there is a substantial reduction in the database size or field width. At this stage, we choose to be ambiguous about this condition by referring to it generally as the *ProjectionCondition.* A new database is created using the subroutine *Project*$(\cdot,\cdot,\cdot)$ of Figure 2. If the database is indeed projected, then the bitvector at node N also needs to be modified in order to reflect this. Since every transaction in this new database is needed in order to generate the frequent extensions of N, we create a new bitvector B' with as many entries as this newly created database. The value of each bit in this database is 1.

Once the bitvectors and newly projected database has been created, we call the algorithm recursively from all frequent extensions of node N. The calls to the frequent extensions of N are made in lexicographic order. These recursive calls create the subtrees which are rooted at the corresponding nodes of the lexicographic tree.

3.1. Heuristic Rules for selective projection

Various heuristic rules may be used in order to decide the exact nature of the projection condition discussed in Figure 2. For example, it is possible to use the heuristic rule that a projected transaction set may be maintained at a node only when the size of the projected database is less than a certain factor of the previous database size. Alternatively, it is possible to maintain a projected transaction set at a node, only when the number of bits to represent the (projected) transaction at that node is less than a certain number *i.e.* $|E(P)|$ falls below a certain number. The primary motivation is to project only when the database representation becomes "sufficiently more efficient" after performing the process. We will discuss later how we chose the projection condition for our implementation. We will also discuss how to combine the depthfirst creation of the lexicographic tree with an effective lookahead strategy in order to avoid creation of those subtrees of the lexicographic tree which contain only non-maximal itemsets.

4. Database Representation

A number of optimizations were performed in order to improve the performance of the algorithm. Since the algorithm discussed in this paper is geared towards generating long patterns, we found that it was better to use the database in the *bitstring representation.* In the bitstring representation, each item in the database has one bit representing it. Thus, the length of each transaction in bits is equal to the total number of items in the database. Such a representation is inefficient when the number of items in each transaction is significantly less than the total number of items. This is because of the fact that most of the bits take on the value of 0. However, the *DepthProject* algorithm performs counting on the projected transactions which are expressed in terms of $E(P)$. In this representation, most bits take on the value of 1. Furthermore, for problems in which the database contains long patterns, the ratio of the maximum pattern length to the total number of items is relatively high for most lower level nodes.

We have discussed the concept of selective projection slightly earlier. In the implementation of the algorithm over several datasets, we found that the best strategy was to project the database only when $|E(P)|$ is reduced to less than 32. In this case, projected transactions can be represented as 32 bit words. Thus, a projected database with one million transactions can be represented in only 4 MB, when $|E(P)|$ is less than 32. This illustrates the considerable advantages of using the bitstring representation for holding the projected transactions. We will now discuss how the bitstring representation is helpful in reducing the counting times substantially.

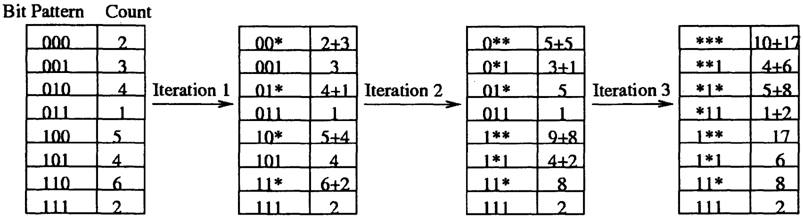
4.1. Counting Methods

Carefully designed counting methods are critical in reducing the times for finding support counts. Let us consider a node P, at which it is desirable to count the support of each item in $F(P)$. Let Q be the nearest ancestor of P at which the projected database $\mathcal{T}(Q)$ is maintained. Each projected transaction in $\mathcal{T}(Q)$ contains n bits, where $n = |E(Q)|$. We are assuming that $|F(P)|$ is close to n. Otherwise, we would have projected the database at some intermediate node between P and Q according to the projection condition in order to reduce the database size. Let T be a transaction in $\mathcal{T}(Q)$. A naive method of counting would be to maintain a counter for each item in $F(P)$ and add one to the counter of each of those elements for which the corresponding bit in T takes on the value of 1. However, it is possible to reduce the counting times greatly by using a two phase byte counting method.

The first step in the counting technique is to decide whether the transaction T is relevant in counting the support of candidate extensions of node P. The transaction T is relevant in counting the support of candidate extensions of node P, if P is a subset of T. The bit vector maintained at the immediate parent P' of P provides the information as to whether the transaction T, was relevant in counting the support of the frequent extensions of P'. If this is so, then we need to check additionally whether the lexicographically largest item of P (this item extends P' to P) is present in T. Once it has been determined that the transaction T is relevant in order to perform the counting at node P, we count the support for each item in $F(P)$. Let us assume that each transaction T contains n bits, and can therefore be expressed in the form of $\lceil n/8 \rceil$ bytes. Each byte of the transaction contains the information about the presence or absence of eight items, and the integer value of the corresponding bitstring can take on any value from 0 to $2^8 - 1 = 255$. Correspondingly, for each byte of the (projected) transaction at a node, we maintain 256 counters, and we add 1 to the counter corresponding to the integer value of that transaction byte. This process is repeated for each transaction in $\mathcal{T}(P)$. Therefore, at the end of this process, we have $256 * \lceil n/8 \rceil$ counts. We follow up with a postprocessing phase in which we determine the support of an item by adding the counts of the $256/2 = 128$ counters which take on the value of 1 for that bit. Thus, this phase requires $128 * n$ operations only, and is independent of database size. The first phase, (which is the bottleneck) is the improvement over the naive counting method, since it performs only 1 operation for each *byte* in the transaction, which contains 8 items. Thus, the method would be a factor of 8 faster than the naive counting technique, which would need to scan the entire bitstring. The counting times are determined by the time spent at the lower level nodes, at which the projected transactions are inherently dense. At the lower levels in the tree, we use another specialized technique called bucketing in order to perform the counting. We will discuss this method in greater detail in the next section.

```
Algorithm AggregateCounts(Input: $2^{|E(P)|}$ buckets with transaction counts)
begin
  { We assume that there are $2^{|E(P)|}$ buckets, one
      corresponding to each bitstring of length $|E(P)|$ }
  $k = |E(P)|$;
  for $i := 1$ to $k$ do
    begin
    for $j := 1$ to $2^k$ do
    if the $i$th bit of bitstring representation of $j$ is 0 then
        begin
        $bucket[j] = bucket[j] + bucket[j + 2^{i-1}]$
        end
    end
end
```

Figure 5. Aggregating bucket counts



Bit Pattern	Count						
000	2	00*	2+3	0**	5+5	***	10+17
001	3	001	3	0*1	3+1	**1	4+6
010	4	01*	4+1	01*	5	*1*	5+8
011	1	011	1	011	1	*11	1+2
100	5	10*	5+4	1**	9+8	1**	17
101	4	101	4	1*1	4+2	1*1	6
110	6	11*	6+2	11*	8	11*	8
111	2	111	2	111	2	111	2

Figure 6. Performing the second phase of bucketing

5. The Bucketing Technique

Most of the nodes in the lexicographic tree correspond to the lower levels. Thus, the counting times at these levels account for most of the CPU times of the algorithm. For these levels, we used a strategy called bucketing in order to substantially improve the counting times. The idea is to change the counting technique at a node in the lexicographic tree, if $|E(P)|$ is less than a certain value. In this case, an upper bound on the number of distinct *projected* transactions is $2^{|E(P)|}$. Thus, for example, when $|E(P)|$ is 9, then there are only 512 distinct projected transactions at the node P. Clearly, this is because the projected database contains several repetitions of the same (projected) transaction. The fact that the number of *distinct* transactions in the projected database is small can be exploited in order to yield substantially more efficient counting algorithms. The aim is to count the support for the entire subtree rooted at P with a quick pass through the data, and an additional postprocessing phase which is independent of database size. The process of performing bucket counting consists of two phases:

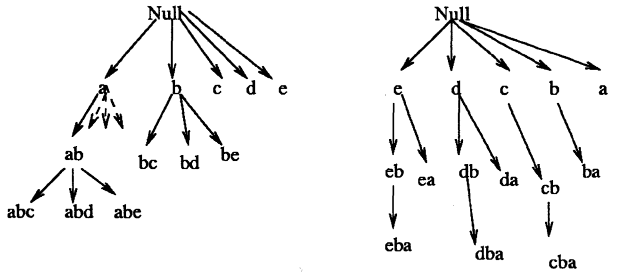
- In the first phase, we count how many of each distinct transaction are present in the projected database. This can be accomplished easily by maintaining $2^{|E(P)|}$ buckets, scanning the transactions one by one, and adding counts to the buckets. The time for performing this set of operations is linear in the number of database transactions.

- In the second phase, we use the $2^{|E(P)|}$ transaction counts in order to determine the aggregate support counts for each itemset. In general, the support count of an itemset may be obtained by adding the counts of all the supersets of that itemset to it. The algorithm for performing these operations is illustrated in Figure 5.

When a string is composed of 0, 1, and $*$, it refers to an itemset in which the positions with 0 and 1 are fixed to those values (corresponding to presence or absence of items), while a position with a $*$ is a "don't care". Thus, all itemsets can be expressed in terms of 1 and $*$, since itemsets are traditionally defined with respect to presence of items. Consider for example, the case when $|E(P)| = 4$, and there are four items, numbered $\{1, 2, 3, 4\}$. An itemset containing items 2 and 4 is denoted by $*1*1$. We start off with the information about $2^4 = 16$ bitstrings which are composed of 0 and 1. These represent all possible distinct transactions. The algorithm aggregates the counts in $|E(P)|$ iterations. The count for a string with a "*" in a particular position may be obtained by adding the counts for the strings with a 0 and 1 in those positions. For example, the count for the string *1*1 may be expressed as the sum of the counts of the strings 01*1 and 11*1.

The procedure in Figure 5 works by starting with the counts of the 0-1 strings, and then converts them to strings with 1 and *. The algorithm requires $|E(P)|$ iterations. In the ith iteration, it increases the counts of all those buckets with a 0 in the ith order bit, so that the count now corresponds to a case when that bucket contains a $*$ in that position. This can be achieved by adding the counts of the buckets with a 0 in the ith order position to that of the bucket with a 1 in that position, with all other bits having the same value. For example, the count of the string 0*1* is obtained by adding the counts of the buckets 001* and 011*. In Figure 5, the process of adding the count of the bucket j to that of the bucket $j + 2^{i-1}$ achieves this.

The second phase of the bucketing operation requires $|E(P)|$ iterations, and each iteration requires $2^{|E(P)|}$ time. Therefore, the total time required by the method is $2^{|E(P)|} \cdot |E(P)|$. When $|E(P)|$ is sufficiently small, the time required by the second phase of postprocessing is small



(a) (b)

Figure 7. Illustrating the effect of using different orderings

compared to the first phase, whereas the first phase is essentially proportional to reading the database for the current projection.

We have illustrated the second phase of bucketing by an example in which $|E(P)| = 3$. The process illustrated in Figure 6 illustrates how the second phase of bucketing is efficiently performed. The exact strings and the corresponding counts in each of the $|E(P)| = 3$ iterations are illustrated. In the first iteration, all those bits with 0 in the lowest order position have their counts added with the count of the bitstring with a 1 in that position. Thus, $2^{|E(P)|-1}$ pairwise addition operations take place during this step. The same process is repeated for two or three more times with the second and third order bits. At the end of three passes, each bucket contains the support count for the appropriate itemset, where the ’0’ for the itemset is replaced by a ”don’t care” which is represented by a ’*’. Note that the number of transactions in this example is 27. This is represented by the entry for the bucket ***. Only 2 transactions contain all three items, which is represented by the bucket 111.

6. Tree Pruning

In order to speed up the performance of the algorithm, we implement *lookaheads*. Consider a node P with a set of prospective branches $F(P)$. If the node $P \cup F(P)$ is a frequent itemset, then it is not necessary to explore the subtree rooted at P. One way of doing this is to parallelize the process of finding the support count for the itemset $P \cup F(P)$ with that of determining the counts of each of the candidate extensions of P. We choose a simpler alternative by using the following method for performing lookaheads. Let $E = \{i_1, \ldots i_{|E|}\}$ be the set of frequent extensions of a node. Each time after calling the *DepthFirst* procedure from the candidate extension i_{r-1} of a node, we check if the set $\{i_{r-1} \ldots i_{|E|}\}$ was determined to be frequent. If so, the candidate extensions for $\{i_r \ldots i_{|E|}\}$ may be pruned from further consideration. This is a slightly weaker version of the pruning procedure, but it can be implemented much more efficiently. The depthfirst technique also provides the ability to quickly discover maximal patterns, and thereby prune away all those branches of the tree such that $P \cup F(P)$ is a subset of some itemset which has already been discovered. This kind of lookahead is far more effective with a depth first strategy of finding patterns, since longer patterns are discovered earlier on in the tree generation process. As a result many subsets of frequent patterns need never be discovered.

The structure of the lexicographic tree is very much dependent upon the lexicographic ordering of the items in the database. For example, consider the case when there are exactly 3 large itemsets: *abc*, *abd*, and *abe*. Let us now consider the cases when the orderings of the items are a, b, c, d, e, and e, d, c, b, a respectively. The lexicographic trees for the two cases are illustrated in Figures 7(a) and (b). As we see, in the case of Figure 7(b), since the branches of the tree at which the maximal itemsets *eba*, *dba*, and *cba* can be found tend to separate out at very high level (level 1), the process of using lookaheads is likely to be more effective in this case. The very first node visited by the depth-first search procedure after the node *null* in the Figure 7(a) is a. In this case the lookahead process $\{a\} \cup \{b, c, d, e\}$ does not yield a frequent itemset. The same is true of the next level-1 node b. On the other hand, in the case of the Figure 7(b), the first node which is visited by the depth-first search procedure is the node e, and the process of lookahead $e \cup \{b, a\}$ yields a frequent itemset. This example tends to suggest the following heuristic rule for choosing good orderings:

The item which occurs in the fewest number of large itemsets hanging at a node should be first and the item occuring in the maximum number of large itemsets should be last.

Unfortunately, we do not know the large itemsets apriori. Consequently, we found the strategy of ordering from least support to most support to be a reasonable approximation of the above goal. This technique has been discussed earlier by other researchers [BAY98] in order to maximize the efficiency of lookaheads.

The efficiency of the algorithm is improved further by using a dynamic ordering as opposed to a static ordering. In the case of dynamic orderings, we reorder the items below each node depending upon the support of each lexicographic tree-extension. We mention here that a recently proposed technique [BCG01] called parent-child pruning can be used to improve the efficiency of the approach even further.

```
Algorithm MakeMaximal(Itemsets : $\mathcal{I}$, BitNumber: k)
begin
  if $\mathcal{I}$ contains one itemset return $\mathcal{I}$
  { Divide $\mathcal{I}$ into two sets
        $\mathcal{I}_1$ and $\mathcal{I}_2$; }
  $(\mathcal{I}_1, \mathcal{I}_2) = Divide(\mathcal{I}, k)$;
  { In the next steps, we make each of $\mathcal{I}_1$
     and $\mathcal{I}_2$ self-maximal}
  $\mathcal{I}_1 = MakeMaximal(\mathcal{I}_1$, k+1);
  $\mathcal{I}_2 = MakeMaximal(\mathcal{I}_2$ k+1);
  $\mathcal{M} = MaximalMerge(\mathcal{I}_1, \mathcal{I}_2)$;
  return($\mathcal{M}$);
end;
```

Figure 8. Creating Maximal Itemsets with the Divide and Conquer Algorithm

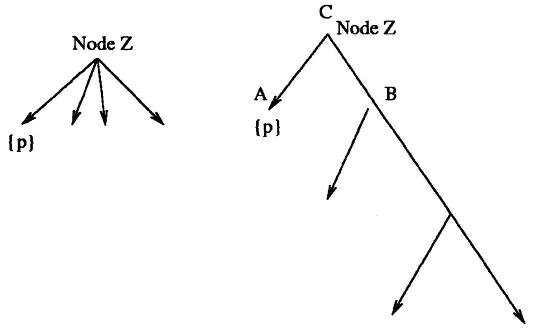


Figure 9. Binarization of the Tree Structure

All itemsets generated in the *DepthProject* algorithm are created by either lookaheads, or by bucketing at the lower level nodes. Therefore, at the termination of the algorithm, we are left with a set of frequent itemsets which contains all frequent maximal itemsets with *some of the* subsets of maximal itemsets. Therefore, at the termination of the *DepthProject* algorithm, all non maximal itemsets are removed by a phase which we refer to as the *post pruning* phase.

6.1. Post Pruning Phase

In the post pruning phase, we find the maximal itemsets with the use of a postprocessing method which uses a divide and conquer approach for separating out the maximal itemsets. Let us assume that the set of itemsets at termination of the *DepthProject* algorithm is denoted by $\mathcal{I}$. In order to facilitate further discussion, we will assume that all itemsets are represented in bit string format. First, we sort all of the itemsets in ascending order of strings. Next, we divide the set of itemsets into two sets $\mathcal{I}_1$ and $\mathcal{I}_2$. This is done using the kth bit of the itemset. At the beginning of the algorithm, we use $k = 1$, which corresponds to the lexicographically least item. If the kth bit is 0, then the itemset belongs to $\mathcal{I}_1$; otherwise it belongs to $\mathcal{I}_2$. The algorithm is applied recursively in order to make $\mathcal{I}_1$ and $\mathcal{I}_2$ self maximal. In the next step $\mathcal{I}_1$ and $\mathcal{I}_2$ are merged, so that the resulting set is also maximal. The merging process may potentially require us to compare every pair of itemsets in $\mathcal{I}_1$ and $\mathcal{I}_2$ in order to make sure that the merged set is maximal. However, this process can be made significantly faster with the use of the fact that the two sets $\mathcal{I}_1$ and $\mathcal{I}_2$ are in sorted order. We know that no itemset in $\mathcal{I}_1$ can be a superset of an itemset in $\mathcal{I}_2$. Therefore, we only need to check whether an itemset in $I^* \in \mathcal{I}_2$ is a superset of an itemset in $\mathcal{I}_1$. In order to achieve this, one might need to scan all itemsets in $\mathcal{I}_1$ in the worst case. The complexity of such a procedure might turn out to be quadratic. The algorithm is illustrated in Figure 8. In the invocation of the algorithm, we use a bit number value of $k = 1$.

The above procedure for merging is a generic one which makes any set of itemsets non-maximal. However, we note that a set of itemsets which are to be made maximal are not arbitrary, but are frequent itemsets, which are discovered in a specific order. We can use this fact in order to improve the efficiency of the post pruning procedure. Let us denote the itemset $I^* \in \mathcal{I}_2$ with the bit flipped at the kth position by I_k^*. We claim that it only suffices to check whether the itemset I_k^* occurs in $\mathcal{I}_1$. This can be accomplished by a simple binary search in $\mathcal{I}_1$, if the itemsets in $\mathcal{I}_1$ are sorted in lexicographic order. In the next paragraph, we will discuss the reasoning behind this fact.

We assume that items are not reordered at intermediate nodes of the lexicographic tree. The original ordering is maintained. Let Z be a node (itemset) in the tree. Note that we can binarize the tree as illustrated in Figure 9. We also note that a depth first search of this binarization corresponds to the order in which the itemsets are discovered by the depth project algorithm. Further, this binarization also corresponds to the implicit tree-structured search by the divide-and-conquer algorithm. We have illustrated this binarization in Figure 9.

Let p be the left most child (item) of Z. Let A be the set of maximal itemsets which include Z and p. And let B be the set of maximal itemsets which include Z but not p. An itemset in A is of the form $(Z, p, X(i))$ while an itemset in B is of the form $(Z, 0, Y(j))$. The goal is to merge A and B to produce set of maximal itemsets C which represents all maximal itemsets at node Z.

Clearly, C may includes all itemsets from A and some from B. An itemset $(Z, 0, Y(j))$ in B is pruned if $Y(j)$ is a subset of some $X(i)$. This will require checking each $Y(j)$ against all $X(i)$'s. This is a quadratic time algorithm. However, we claim that it is sufficient to check for equality. $Y(j)$ is pruned if and only if it is equal to some $X(i)$.

Let $X(i)$ be a superset of $Y(j)$. Then we claim that set B also includes $(Z, 0, W)$, where W is some superset of $X(i)$. Because set B is a maximal set for all itemsets which include Z and not p. If $(Z, p, X(i))$ is a maximal itemset in the left branch of the tree then $(Z, 0, W)$ must be a maximal itemset in the right branch of the tree, where $W \supseteq X(i)$. Thus (Z,0,W) is included in set B. Since $W \supseteq X(i) \supseteq Y(j)$, $(Z, 0, W)$ becomes a superset of $(Z, 0, Y(j))$ and therefore, $(Z, 0, Y(j))$ is not really maximal with respect to set B.

Thus if sets A and B are truly maximal, then $(B, 0, Y(j))$ is pruned if and only if $Y(j) = X(i)$ for some i. We order sets A and B, and using a linear scan, we prune away those Y's which are exactly equal to some X. If we do not reorder items at a node, then sorting needs to be done only once.

Our proof requires using an initial value of $k = 1$, which corresponds to the lexicographically least item, or the top of the binarized version of the tree. We merge the two maximal itemsets corresponding to the presence/absence of the first item. The algorithm is called recursively till we reach the bottom of the tree. This can result in a large overhead associated with keeping track of the small number of itemsets at the bottom of the tree. Note that any recursive algorithm can be converted to a non-recursive version using a process discussed in [AHU74]. In this case, this corresponds to starting at the bottom of the binarized tree and working our way back up. We simulateneously work on all itemsets $\mathcal{I}$ in the tree at one time. We sort the set $\mathcal{I}$ and start with $k = M$ (last item) and work back towards $k = 1$. That will require M passes through the itemsets. In the kth pass, we merge maximal itemsets of the form $(Z, k, X(j))$ with $(Z, 0, Y(j))$, where Z represents an itemset involving items 1 through $k - 1$, and $X(i)$ and $Y(j)$ represent itemset involving items from $k + 1$ to M.

The complexity is $N * M + N * log(N)$, where M is the number of items, and N is the number of itemsets. If we reorder items,then we have to remember it and do the merging in the same order. In this case the worst case complexity is $N * M * Log(N)$, because we have to sort itemsets at every merge step and for an itemset we have at the most M merge steps.

Data Set	**Items**	**Avg. Wdth**	**Recs.**	**Db. Sz. (MB)**	**$\|L_1\|$ (Lowest Support)**	**Lngst. Pat. (Least Sup.)**
con.-4	130	43	67557	12	73	29
mush.	120	23	8124	0.85	115	21
chess	76	37	3196	0.51	54	23
Pumsb	7117	74	49046	10	147	63

Table 1. Illustrating the features of the Data Sets

6.2. Space Requirements

The space requirements of the depth first strategy are not significantly more than the size of the transaction set itself. This is because only one path of the lexicographic tree is being explored at any moment of time. The space required for maintaining all the sets of bitvectors is a small fraction of the transaction set size. If the geometric reduction rule for selective projection is used, and the database is projected only if it reduces by a factor of $Q > 1$, then the space requirements for all transaction sets on the current path are no larger than $Q/(Q-1)$ of the original transaction set size. Furthermore, we have already shown that for problems which are amenable to the depth first strategy, the transaction representation can be changed to a form which is much more efficient. This results in considerable savings.

7. Empirical Results

The experiments were performed on an IBM RS/6000 43P-140 workstation with a CPU clock rate of 200 MHz, 128 MB of main memory, 1MB of L2 Cache and running AIX 4.1. The data resided in the AIX file system and was stored on a 2GB SCSI drive.

We tested the algorithm on a number of datasets which have been used earlier [BAY98], for testing the algorithm in the case when it is possible to mine long patterns from the data. The algorithm proposed in [BAY98] is very efficient for the case of finding long maximal patterns because of its ability to mine out the max-patterns by using look-ahead techniques.

The characteristics of the data sets are illustrated in the Table 1. The next to last column in this table illustrates the number of large 1-itemsets at the lowest support level. As we can see, the ratio of the average record width to $|L_1|$ in these data sets is quite high compared to typical retail data. This characteristic is true for most datasets which have long patterns in them. This ratio increases significantly at the

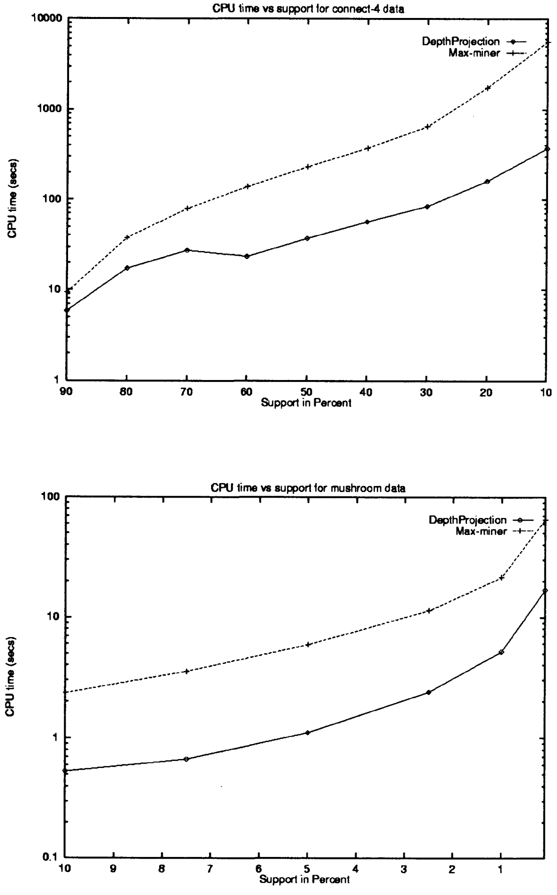


Figure 10. Computational performance on the (a) *connect-4* and (b) *mushroom* data sets

lower level nodes, when the database is projected. The length of the longest pattern at the lowest support level is in the last column of the Table 1. As we can see, many of these patterns are so long that explicit enumeration of all subsets of these patterns would have been unrealistic.

In Figures 10(a) and 10(b), we have illustrated the performance of our algorithm versus the *MaxMiner* method on the *connect-4* and *mushroom* datasets. All plots are made on a logarithmic scale in order to illustrate the order-of-magnitude performance difference between the different algorithms. As we see, the *DepthProject* algorithm quickly outstrips the *MaxMiner* method when support values are low. For the *connect-4*

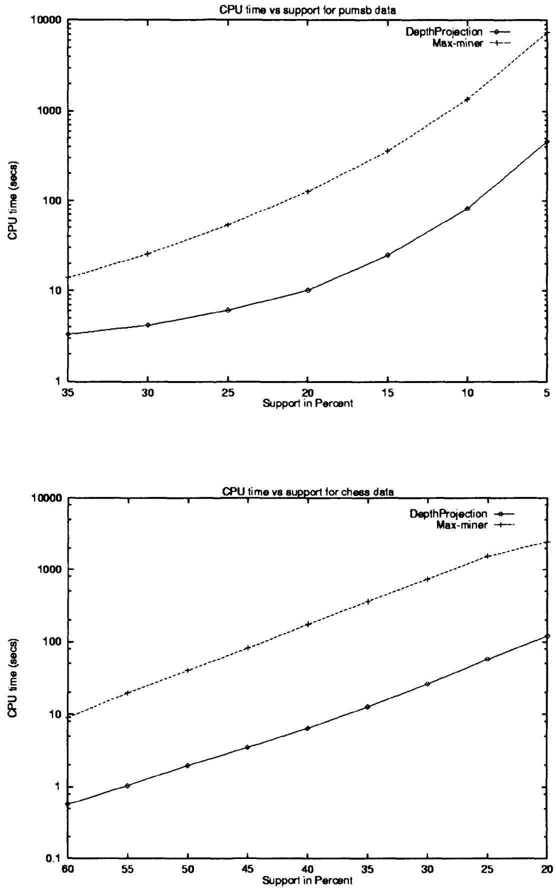


Figure 11. Computational performance on the (a) *pumsb* and (b) *chess* data sets

Data Set	$T(-32)$	$T(+32)$	Cntng	Pruning	Itemsets Found	Maximals
connect-4	7.74	347.55	355.29	8.62	131420	130986
mushroom	0.88	10.30	11.18	5.55	70321	27712
chess	0.13	89.84	89.97	30.58	589495	509355
Pumsb	21.98	401.55	423.53	33.48	313268	98140

Table 2. Running Time in various phases of the algorithm at the lowest support level

dataset, at the highest support value of 90%, the *MaxMiner* algorithm required 9.4 seconds, while the *DepthProject* technique required 5.87 seconds. At the lowest support value of 10%, the *MaxMiner* algorithm required 5624.7 seconds, while the *DepthProject* algorithm required only 370.38 seconds. Thus, the performance gap increased at lower values of support, so that at the lowest support value, the performance gap between the two methods was a factor of 15.2.

The mushroom dataset also exhibits a large performance gap, though the difference is less dramatic in this case. At the highest support level of 10%, the *MaxMiner* algorithm required 2.35 seconds, whereas the *DepthProject* algorithm required 0.53 seconds. At the lowest support level[4] of 0.1%, *MaxMiner* algorithm required 65.19 seconds, whereas the *DepthProject* algorithm required 16.92 seconds. The performance gap between the two methods was a factor of 4 to 5 for all the support levels tested.

The most striking performance improvements were observed in the case of the *chess* and *pumsb* data sets. This is because these data sets correspond to cases when there are a large number of very long patterns. The performance numbers for the *pumsb* dataset are illustrated in Figure 11(a). At the highest support level of 35%, the *MaxMiner* algorithm required 13.91 seconds, whereas the *DepthProject* algorithm required 3.31 seconds. Thus, the performance gap at the highest support level was a factor of 4. At the lowest support level of 5%, *MaxMiner* algorithm required 461.19 seconds, while the *DepthProject* algorithm required 7378.6 seconds. This is a performance gap of approximately 16.

In the case of the *chess* dataset, which is illustrated in Figure 11(b), the most dramatic differences in performance were observed. At the highest support level of 60%, the *MaxMiner* algorithm required 9.08 seconds, whereas the *DepthProject* algorithm required 0.58 seconds. This is a performance gap of approximately 15.655. At the lowest support level of 20%, the *MaxMiner* algorithm required 2442.58 seconds, whereas the *DepthProject* algorithm required 120.06 seconds. Thus, at the lowest support level, the performance gap was a factor of 20.34.

The *DepthProject* algorithm required a postprocessing phase in which the non-maximal itemsets were removed. (The times illustrated include the postprocessing phase.) The time for this postprocessing phase depended upon the number of itemsets which were found by the tree generation and counting phase of the algorithm. As we can see from the the last two columns of Table 2, the number of itemsets found closely matches the maximal itemsets for the *chess* and *connect-4* data sets. This indicates that the process of lookaheads was quite effective in removing the non-maximal patterns quickly. The times for counting and pruning are also illustrated in the same table. (The column corresponding to the counting procedure may be obtained by adding the $T(+32)$ and $T(-32)$ columns, which will be explained below.)

The implementation of the *DepthProject* algorithm assumed that the only time that a database was projected at a node was at the highest level of the tree at which the number of frequent extensions of the itemset became less than 32. It is instructive to compare the time taken in order to generate above the level of 32 bit projection to the time taken in order to generate the tree below the level of 32 bit projection. It is apparent from Table 2 that the time required in order to count below the 32 bit level ($T(+32)$) was substantially more than the time required to count above the 32 bit level ($T(-32)$) for all four data sets. Furthermore, since the most efficient counting of the algorithm on a *per node basis* was performed below the 32 bit level, it follows from the time distributions of the two cases that most of the nodes on which the counting was performed were below the 32 bit level. This also establishes the fact that the efficient representation of the database at the lower levels of the tree contribute substantially to the savings in computational time.

8. Conclusions and Summary

This paper investigated a novel technique for mining long patterns in databases, which relies on a depth-first search technique in order to construct the lexicographic tree of itemsets. The use of depth-first search in reducing the number of itemsets is critical in being able to mine very long patterns at low values of support. The technique of depth-first search is optimum from the perspective of node pruning and performing the counting efficiently by using a set of projected transactions, as opposed to the transactions themselves. The depth first tree generation technique provides the ability to generate long patterns of the tree fast, and prune away the non-maximal patterns earlier on the the process. At the same time, it provides the ability to reuse the information from counting k-patterns in order to count the support of $(k+1)$-patterns. These characteristics explain the excellent computational properties of the *DepthProject* algorithm.

9. Acknowledgements

We would like to thank Rakesh Agrawal and Roberto Bayardo for providing us with several datasets for testing our algorithms. Roberto also provided us with access to his *MaxMiner* code for performing experiments.

Notes

1. This is an extended version of the paper [AAV00] published in the ACM SIGKDD Conference, 2001.

2. If desired, the main memory algorithm in this paper can be combined with a method discussed in [SON95] which divides a disk-resident database into different main-memory partitions and then combines the itemsets from the different partitions in order to generate the final itemsets. This is however not the focus of this paper.

3. In the implementation section, we shall describe in detail which nodes are the ones at which the projected transaction sets are maintained.

4. Note that the lowest support level for Figure 10(b) is 0.1%.

References

[AAV01] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A Tree Projection Algorithm for finding frequent itemsets. *Journal on Parallel and Distributed Computing*, 2001.

[AAV00] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth First Generation of Long Patterns. *Proceedings of the ACM SIGKDD Conference*, 2000.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in very large databases. *ACM SIGMOD Conference*, 1993.

[AMS94] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. *Advances in KDD.* U. M. Fayyad et. al. (Eds.) AAAI/MIT Press, Chapter 12, pages 307-328. *VLDB Conference,* pages 478-499, 1994.

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. The Design and Analysis of Computer Algorithms. *Addison Wesley Publishing Company*, 1974.

[BAY98] R. J. Bayardo. Efficiently Mining Long Patterns from Databases. *ACM SIGMOD Conference,* 1998.

[BCG01] D. Burdick, M. Calimlim, J. Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm For Transactional Databases. *Proceedings of the ICDE Conference*, 2001.

[BMUT97] S. Brin, R. Motwani, J. D. Ullman and S. Tsur. Dynamic Itemset Counting and implication rules for Market Basket Data. *ACM SIGMOD Conference*, 1997.

[SMB97] C. Silverstein, R. Motwani, S. Brin. Beyond Market Baskets: Generalizing association rules to correlations. *ACM SIGMOD Conference*, 1997.

[DS99] B. Dunkel, N. Soparkar. Data organization and access for efficient data mining. *ICDE Conference*, 1999.

[GMS97] D. Gunopulos, H. Mannila, S. Saluja. Discovering All Most Specific Sentences by Randomized Algorithms. *ICDT*, 1997, 215-229.

[HPY00] J. Han, J. Pei, Y. Yin. Mining Frequent Patterns without Candidate Generation. *ACM SIGMOD Conference*, 2000.

[HTV94] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. *AAAI Workshop on KDD,* 1994.

[LNHP99] L. V. S. Lakshmanan, R. Ng, J. Han, A. Pang. Optimization of Constrained Frequent Set Queries with 2-variable Constraints. *ACM SIGMOD Conference*, 1999.

[RF98] I. Rigoutsos, A. Floratos. Combinatorial Pattern Discovery in Biological Sequences. *Bioinformatics*, Volume 14, no 1, 1998, pages 55-67.

[SON95] A. Savasere, E. Omiecinski, S. B. Navathe. An efficient algorithm for mining association rules in large databases. *VLDB Conference*, 1995.

[SA95] R. Srikant, R. Agrawal. Mining Generalized Association Rules. *VLDB Conference*, 1995 pages 407-419.

[SA96] R. Srikant, R. Agrawal. Mining quantitative association rules in large relational tables. *ACM SIGMOD Conference*, 1996.

[TO96] H. Toivonen. Sampling Large Databases for Association Rules. *VLDB Conference*, 1996.

[ZPOL97] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li. New Algorithms for Fast Discovery of Association Rules, *KDD*, Newport Beach, CA, USA, 14-17, August 1997.

Chapter 29

PROBABILISTIC ESTIMATION IN DATA MINING

Edwin P. D. Pednault and Chidanand Apte

Abstract The goal of scientific inquiry is to uncover the principles that govern the world around us, and ultimately to express those principles in a mathematical form that reflects the empirical characteristics of observed data. In this regard, we have been exploring ways of modifying machine learning techniques so that the resulting predictive models likewise reflect the empirical characteristics of observed data. Following the principles of robust estimation, our methodology involves first examining the data to identify an appropriate family of statistical distributions for modeling the data, and then incorporating the corresponding maximum-likelihood estimation procedures into a decision tree algorithm. We have applied this methodology to insurance risk modeling and have obtained tree-based models superior to those obtained using conventional classification and regression tree algorithms.

Keywords: Predictive modeling, insurance risk modeling, robust estimation, decision trees, classification and regression trees

1. Introduction

The goal of scientific inquiry is to uncover the principles that govern the world around us, and ultimately to express those principles in mathematical form. The discovery process involves hypothesizing, testing, and refining potential scientific principles using both theoretical and experimental methods. Through this process, principles are eventually discovered and formulated mathematically in such a way as to reflect the empirical characteristics of observed data.

We have been exploring ways of modifying machine learning techniques so that the resulting predictive models likewise reflect the empirical characteristics of observed data. In particular, we are interested in

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 567–589.
© 2001 *Kluwer Academic Publishers.*

constructing models that not only predict the most probable outcome, but also predict the probability distribution over possible outcomes.

Our work is highly influenced by results obtained in robust estimation [R.R97]. Conventional modeling techniques are often based on certain fixed statistical assumptions; for example, on assumptions of normality in the case of least-squares regression. When the observed data violates the assumptions that underlie the modeling techniques being employed, the resulting models can be unreliable. To obtain reliable models, the statistical assumptions of the modeling techniques must be consistent with the statistical properties of the data being modeled.

Figure 1a illustrates some of the problems that can arise when the statistical assumptions of a modeling technique do not reflect the empirical characteristics of the data. In this example, least-squares regression is used to model a linear relationship in which the distribution of the residuals is heavy-tailed—that is, most of the data points are concentrated near a straight line that relates X to Y; however, there is also a substantial wide-spread background scatter of data points that lie quite far from the line. When least-squares regression is used to estimate the coefficients of the linear relationship, the points in the background scatter have an undue influence on the estimates of the regression coefficients. Consequently, the coefficient estimates are highly inaccurate, and so too is the resulting model.

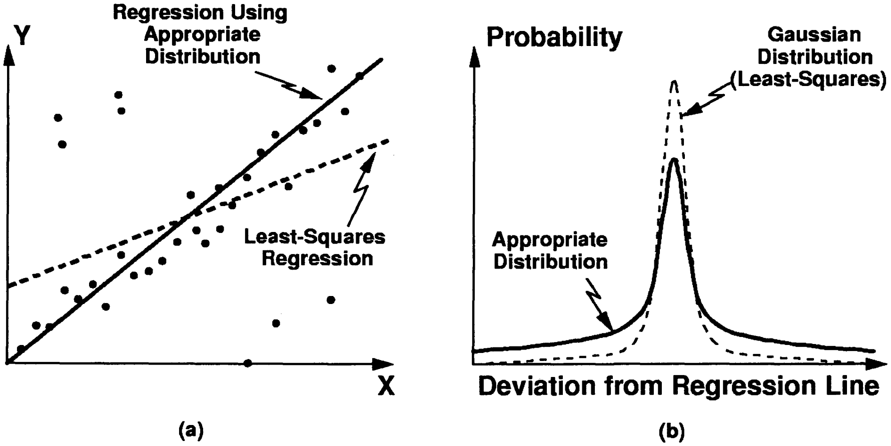


Figure 1. A basic principle of robust estimation is that the accuracy of regression models can be maximized by modeling the residuals using mathematical distributions that accurately reflect the observed distributions of the data.

This inaccuracy is due largely to the fact that the extent of the background scatter violates the statistical assumptions that underlie least-squares regression. Least-squares regression implicitly assumes that the residuals are normally distributed, whereas the actual distribution of the residuals in this example is highly non-normal (i.e., non-Gaussian). Least-squares methods estimate regression coefficients by adjusting the coefficients so as to maximize the probability density of the residuals assuming that the residuals obey a normal distribution. According to a normal distribution, the probability density rapidly falls to zero for residuals that deviate more than two standard deviations from the regression line (see Figure 1b). If the actual probability density of the residuals does not fall as rapidly to zero as does a normal distribution, then the contributions of distant data points to the overall likelihood of the data will begin to dominate the contributions of closer data points. As the contributions of extreme data points begin to dominate, the regression line will adjust so as to reduce the distance from the regression line to the extreme points. The adjustment occurs in order to maximize the overall likelihood of the data, assuming a Gaussian noise model. The resulting effect can be observed in Figure 1a.

Some methods of robust estimation involve deleting extreme values by treating them as "outliers." However, such methods are analogous to assuming that extreme values represent erroneous data. Deletion of these values is then justified as a data cleansing step to eliminate erroneous values from consideration.

Other methods of robust estimation are based on the use of probability distributions that better reflect the observed distribution of the data. This approach assumes that extreme values are legitimate data values—that is, extreme values are actually observed in practice and are not in any way erroneous. Consequently, the statistical assumptions of the modeling technique must be adjusted so as to take into account (i.e., model) the extreme values that might be present.

This latter approach is the one we have adopted in our work. We presume that the data have already been cleansed of erroneous values, and that any remaining extreme values are legitimate data values actually observed in practice.

When applied to linear regression, this latter approach to robust estimation requires that a suitable probability distribution be selected that reasonably approximates the empirical distribution observed in the residuals. In the case of the example shown in Figure 1a, this means selecting a heavy-tailed distribution such as the one shown in Figure 1b. When the latter distribution is used to obtain maximum-likelihood estimates of the regression coefficients, the resulting estimates produce a much more reasonable regression line as shown in Figure 1a. The resulting estimation procedure is not a least-squares procedure because it does not assume a normal distribution; however, it is still a maximum-likelihood procedure relative to a different distributional assumption.

In applying robust estimation in the context of machine learning, we have been examining extensions to decision-tree methods that allow arbitrary distributional assumptions to be incorporated into the models used in the leaves of trees. Specifically, in place of the usual classification model or Gaussian (i.e., squared-error) model, we have instead been investigating the use of more sophisticated statistical models.

Our first effort in extending decision tree methods has focused on modeling insurance risks for personal-lines (i.e., non-commercial) property and casualty insurance. The modeling technique we developed uses joint Poisson/lognormal statistical models in the leaves of decision trees to simultaneously model the frequency with which insurance claims are filed and the amounts of those claims. The choice of a joint Poisson/lognormal statistical model was made after an examination of the data to be modeled, following the principles of robust estimation. Joint Poisson/lognormal statistical models have other uses besides insurance risk modeling, and can be generally used to model random processes consisting of events that occur over time and that have positive quantities associated with those events.

The methodology we developed also demonstrates how other types of statistical models can be incorporated into the leaves of trees. Our methodology involves first examining the data to identify an appropriate family of distributions to employ, and then incorporating the corresponding maximum-likelihood estimation procedures into a decision tree algorithm. Maximum-likelihood principles are used not only to estimate the parameters of the statistical models that appear in the leaves of the trees, but also to select splitting factors during tree-building, and to prune overfitted trees after tree-building is complete.

2. Insurance Risk Modeling

The property and casualty (P&C) insurance business deals with the insuring of tangible assets, such as cars, boats, homes, etc. The insuring company evaluates the risk of the asset being insured, taking into account characteristics of the asset as well as the owner of the asset. Based on the level of risk, the company charges a certain fixed, regular premium to the insured. Actuarial analysis of policy and claims data plays a major role in the analysis, identification, and pricing of P&C risks.

Actuaries develop insurance risk models by segmenting large populations of policies into predictively accurate risk groups, each with its own distinct risk characteristics. A well-known segment is male drivers under age 25 who drive sports cars. Examples of risk characteristics include mean claim rate, mean claim severity amount, pure premium (i.e., claim rate times severity), and loss ratio (i.e., pure premium over premium charged). Pure premium is perhaps the most important risk characteristic because it represents the minimum amount that policyholders in a risk group must be charged in order to cover the claims generated by that risk group. Actual premiums charged are ultimately determined based on the pure premiums of each risk group, as well as on the cost structure of the insurance company, its marketing strategy, competitive factors, etc.

Insurance risk modeling can be approached as a machine learning problem; however, one is then faced with several challenges. One challenge is that specialized, domain-specific equations must be used to estimate claim frequency and claim severity. Another challenge is that some types of claims can take several years to settle, most notably bodily injury claims. Consequently, the claim amount can be a missing value. A suitable learning algorithm would therefore have to compensate for such "partially labeled" training examples. A third challenge is that insurance actuaries demand tight confidence bounds on the risk parameters that are obtained; that is, the risk groups identified by the learning algorithm must be *actuarially credible.*

The above challenges imply that standard machine learning algorithms are ill-suited for this application. If one were forced to use standard algorithms such as CHAID [Kas80, BdS80], CART [BFOS84], C4.5 [Qui93], or SPRINT [SAM96], one might try to view frequency modeling as a classification problem and severity modeling as a regression problem. However, further examination suggests that these modeling tasks are unlike standard classification or regression problems. Viewing frequency prediction as a classification problem is misleading. It is certainly not the case that every individual policyholder will file a claim with either 100% certainty or 0% certainty. In actuality, every individual has the potential to file claims, it is just that some do so at much higher rates than others. The predictive modeling task is therefore to discover and describe different groups of policyholders, each with their own unique filing rates, rather than attempt to discover groups that are "classified" as either always filing claims or never filing claims.

Frequency prediction is also not the same as predicting class probabilities. Claim frequency is the average rate at which individual policyholders from a risk group file for claims. Frequency is usually expressed as the number of claims filed per policy per unit time (i.e., quarterly, annually, etc.); however, it can also be expressed as a percentage by multiplying by 100. For example, a frequency of 25% means that the average number of claims filed in a given unit of time is 0.25 times the number of policies. This is not to say that 25% of policyholders file claims; only about 19.5% will file one claim in the given time period and an unlucky 2.6% will file two or more claims. Thus, the 25% refers to a rate, not a probability.

Severity prediction, on the other hand, is very much a regression problem, given that the data fields that correspond to this variable have continuous values across a wide range. However, the distributional characteristics of claim amounts are quite different from the traditional Gaussian (i.e., least-squares optimality) assumption incorporated into most regression modeling systems. Insurance actuaries have long recognized that the severity distribution is often highly skewed with long thick tails. Reliance on the Gaussian assumption for modeling individual claims can lead to suboptimal results, which is a well-known problem from the point of view of robust estimation, as previously discussed.

The above challenges have motivated our own research [HPS97, Ped99] and have lead to the development of the IBM ProbE$^{\mathrm{TM}}$ (*Prob*abilistic *E*stimation) predictive modeling kernel. This C++ kernel embodies several innovations that address the challenges posed by insurance data. The algorithms are able to construct rigorous rule-based models of insurance risk, where each rule represents a risk group. The algorithms differ from standard data mining algorithms in that the domain-specific calculations necessary for modeling insurance risks are not only integrated into the algorithms, they are in fact used to help guide the search for risk groups.

3. Statistical Modeling of Insurance Risks

Actuarial science is based on the construction and analysis of statistical models that describe the process by which claims are filed by policyholders (see, for example, [KPW98]). Different types of insurance often require the use of different statistical models. The statistical models that are incorporated into the current version of ProbE are geared toward property and casualty insurance in general, and automobile insurance in particular.

For any type of insurance, the choice of statistical model is often dictated by the fundamental nature of the claims process. For property and casualty insurance, the claims process consists of claims being filed by policyholders at varying points in time and for varying amounts. In the normal course of events, wherein claims are not the result of natural disasters or other widespread catastrophes, loss events that result in claims (i.e., accidents, fire, theft, etc.) tend to be randomly distributed in time with no significant pattern to the occurrence of those events from the point of view of insurable risk (see Figure 2). Policyholders can also file multiple claims for the same type of loss over the life of a policy.

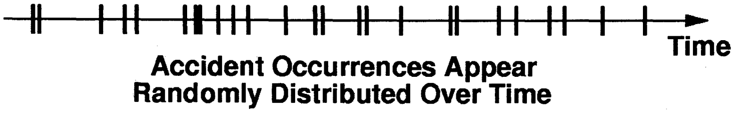


Figure 2. A defining characteristic of Poisson random processes is that events occurrences are distributed randomly over time.

Claim filings such as these can be modeled as a Poisson random process [KPW98], which is the appropriate mathematical model for events that are randomly distributed over time with the ability for events to reoccur (i.e., renew). The Poisson model is likewise used in the current version of ProbE.

It is important to note that Poisson processes are not appropriate for modeling catastrophic claims arising from widespread disasters—such as hurricanes, earthquakes, floods, and so forth—because such catastrophes do not produce uniform distributions of loss events over time. Instead, natural disasters give rise to clusters of claims being filed over short periods of time, where the number of claims in each cluster depends both on the number of policyholders in the region affected by the disaster and on the severity of the disaster. An appropriate statistical model should likewise take into account the geographic distribution of policyholders relative to the damage caused by disasters. Such a model would be substantially more sophisticated than the simple Poisson model currently used in ProbE, and it would have to be incorporated into ProbE in order to adequately model the true levels of risk posed by widespread catastrophes.

The Poisson model is also not appropriate for modeling other forms of insurance, such as life insurance in which at most one death benefit is ever claimed per policy. Life insurance is best modeled as a survival process, not a renewal process. Additional probabilistic models would have to be incorporated into ProbE to model such forms of insurance. Such extensions, however, should be relatively easy to accomplish compared to the extensions that would be needed to model catastrophic losses.

In addition to modeling the distribution of claims over time, actuaries must also model the amounts of those claims. In actuarial science, claim amounts for property and casualty insurance are modeled as probability distributions. Two kinds of distributions are usually considered:

those for the amounts of individual claims, and those for the aggregate amounts of groups of claims. In principle, aggregate loss distributions can be derived mathematically from the distributions of the individual losses that make up the sum. However, only in a few special cases can closed-form solutions be obtained for these mathematical equations. In most cases, approximations must be employed. Fortunately, actuaries typically consider large groups of claims when analyzing aggregate loss. The central limit theorem can therefore be invoked and aggregate losses can be reasonably approximated by normal (i.e., Gaussian) distributions.

The distributions incorporated into the current version of ProbE were selected based on an examination of historical automobile claims data. Claim amounts were found to have highly skewed distributions in this data. Most claims were small in value relative to the maximum amounts covered by the policies, but a significant proportion of large claims were also present. When the claim amounts were logarithmically transformed, the skewness virtually disappeared and the resulting distributions were found to be highly Gaussian in shape. These properties are the defining characteristics of lognormal distributions, an example of which is illustrated in Figure 3. Other statistical distributions employed by actuaries (see, for example, [KPW98]) can also be incorporated into ProbE and used in place of the lognormal distribution; however, the lognormal distribution was found to provide the best overall fit to the observed distribution of claim amounts in the available data (the degree of fit was assessed using quantile-quantile (Q-Q) plots, a standard visual method used in applied statistics for assessing the degree of fit between an assumed theoretical distribution and the empirical distribution of a set of data points). The lognormal distribution was therefore selected as the first distribution to be incorporated into ProbE for risk modeling purposes.

Unfortunately, there are no closed-form solutions for the aggregate loss distribution given that individual losses follow a lognormal distribution. In particular, a sum of lognormal random variables is not itself lognormal. An approximation must therefore be made. In ProbE, the central limit theorem is invoked and the normal distribution is used to model aggregate losses. It should be noted, however, that aggregate loss distributions are not used by ProbE during data mining, but only for post-mining analysis when estimating the aggregate parameters of each risk group. The above approximation therefore has no effect on the risk groups that are identified by ProbE.

Because different distributions are used to model individual versus aggregate losses, different statistical procedures are employed for estimating the parameters of these distributions. For the lognormal distribu-

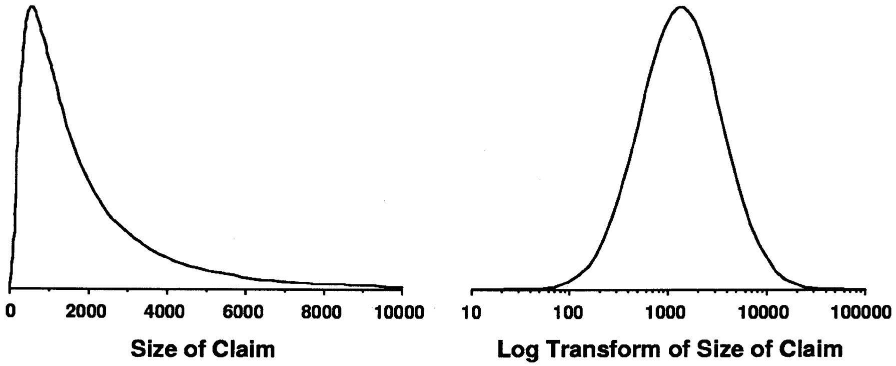


Figure 3. A defining characteristic of lognormal random variables is that their logarithms are normally distributed.

tions used to model individual losses, the relevant statistical parameters are the means and standard deviations of the natural logarithms of the individual claim amounts. For the normal distributions used to model aggregate losses, the means and standard deviations of the (raw) claim amounts are the parameters that need to be estimated.

4. Top Down Identification of Risk Groups

The traditional method used by actuaries to construct risk models involves first segmenting the overall population of policyholders into a collection of risk groups based on a set of factors, such as age, gender, driving distance to place of employment, etc. The risk parameters of each group are then estimated from historical policy and claims data. Ideally, the resulting risk groups should be homogeneous with respect to risk; that is, further subdividing the risk groups by introducing additional factors should yield substantially the same risk parameters. Actuaries typically employ a combination of intuition, guesswork, and trial-and-error hypothesis testing to identify suitable factors. The human effort involved is often quite high and good risk models can take several years to develop and refine.

ProbE replaces manual exploration of potential risk factors with automated search. Risk groups are identified in a top-down fashion by a method similar to those employed in classification and regression tree algorithms [BFOS84, Kas80, BdS80, SAM96, Qui93]. Starting with an overall population of policyholders, ProbE recursively divides the policyholders into risk groups by identifying a sequence of factors that produce the greatest increase in homogeneity within the subgroups that are produced. The process is continued until each of the resulting risk groups is either declared to be homogeneous or is too small to be further subdivided from the point of view of actuarial credibility.

One of the key differences between ProbE and other classification and regression tree algorithms is that splitting factors are selected based on application-specific statistical models that appear in the leaves of the trees. In the case of insurance risk modeling, a joint Poisson/lognormal model is used to enable the simultaneous modeling of frequency and severity and, hence, pure premium. The joint Poisson/lognormal model explicitly takes into account insurance-specific variables, such as earned exposure and claim status. In addition, it provides feedback to ProbE's search engine on the degree of actuarial credibility of each proposed risk group so that only those splitting factors that yield actuarially credible risk groups are considered for further exploration. By explicitly taking these aspects of the problem into account, ProbE is able to overcome the major barriers that cause standard data mining algorithms to be suboptimal for this application.

5. The Joint Poisson/Lognormal Model

The optimization criterion used to identify splitting factors is based on the principles of maximum likelihood estimation. Specifically, the negative log-likelihood of each data record is calculated assuming a joint Poisson/lognormal statistical model, and these negative log likelihoods are then summed to yield the numerical criterion that is to be optimized. Minimizing this negative log-likelihood criterion causes splitting factors to be selected that maximize the likelihood of the observed data given the joint Poisson/lognormal models of each of the resulting risk groups.

Historical data for each policy is divided into distinct time intervals for the purpose of data mining, with one data record constructed per policy per time interval. Time-varying risk characteristics are then assumed to remain constant within each time interval; that is, for all intents and purposes their values are assumed to change only from one time interval to the next. The choice of time scale is dictated by the extent to which this assumption is appropriate given the type of insurance being considered and the business practices of the insurer. For convenience, quarterly intervals will be assumed to help make the discussion below more concrete, but it should be noted that monthly or yearly intervals are also possible

Assuming that data is divided into quarterly intervals, most data records will span entire quarters, but some will not. In particular, data records that span less than a full quarter must be created for policies that were initiated or terminated mid-quarter, or that experienced mid-quarter changes in their risk characteristics. In the case of the latter, policy-quarters must be divided into shorter time intervals so that separate data records are created for each change in the risk characteristics of a policy. This subdivision must be performed in order to maintain the assumption that risk characteristics remain constant within the time intervals represented by each data record. In particular, subdivision must occur when claims are filed under a policy in a given quarter because the filing of a claim can itself be an indicator of future risk (i.e., the more claims one files, the more likely one is to file future claims). The actual time period covered by a database record is the earned exposure of that record.

Figure 4 depicts the database records that are constructed as a result of subdivision. In this figure, Q0, Q1, Q2, etc., represent the ending days of a sequence of quarters. T0 represents the day on which a particular policy came into force, while T1 represents the day the first claim was filed under that policy. Though not illustrated, T2, T3, T4, etc., would represent the days on which subsequent claims were filed. For data mining purposes, the policy claims data is divided into a sequence of database records with earned exposures t1, t2, t3, etc.

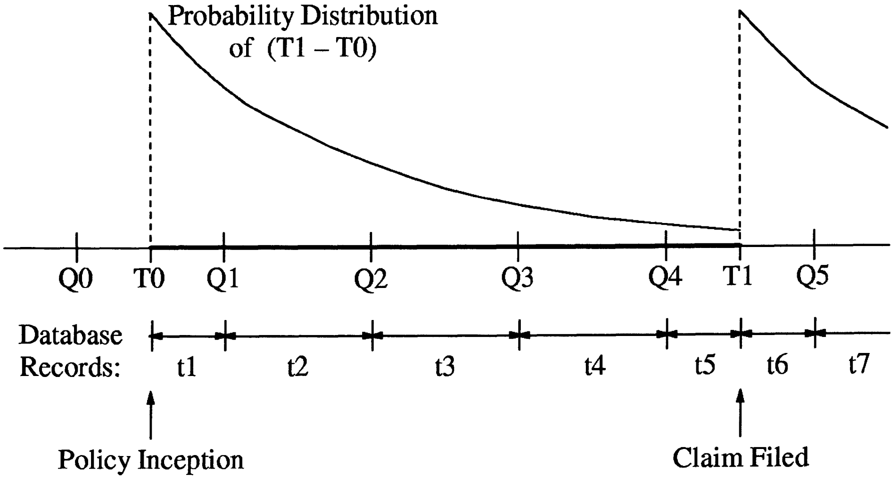


Figure 4. Policy data must be divided into time intervals that are short enough to support the assumption that risk characteristics remain constant within each interval.

As illustrated, new policies typically come into force in the middle of quarters. Thus, the earned exposure for the first quarter of a policy's existence (e.g., t1) is generally less than a full quarter. The earned exposures for subsequent quarters, on the other hand, correspond to full quarters (e.g., t2, t3, and t4) until such time that a claim is filed, the risk characteristics change mid-quarter, or the policy is terminated. When a claim is filed or the risk characteristic changes, the data for that quarter is divided into two or more records. The earned exposure for the first database record (e.g., t5) indicates the point in the quarter at which the claim was filed. The earned exposure for the second record (e.g., t6) indicates the time remaining in the quarter, assuming only one claim is filed in the quarter as illustrated in the diagram. If two or more claims are filed in the quarter, then three or more database records are constructed: one record for each claim and one record for the remainder of the quarter (assuming that the policy has not been terminated). Likewise for other changes in risk characteristics, such as adding or removing drivers, cars, etc., from the policy.

For Poisson random processes, the time between claim events follows an exponential distribution. Moreover, no matter at what point one starts observing the process, the time to the next claim event has the same exponential distribution as the time between claim events. For the example shown in Figure 4, the probability density function for the time between the policy inception and the first claim being filed is given by

$$f(\mathrm{T1}-\mathrm{T0}) = \lambda e^{-\lambda(\mathrm{T1}-\mathrm{T0})} = \lambda e^{-\lambda(\mathrm{t1}+\mathrm{t2}+\mathrm{t3}+\mathrm{t4}+\mathrm{t5})}\,, \tag{29.1}$$

where λ is the claim frequency of the risk group. More generally, from the additivity properties of Poisson random processes, it can be shown that the probability density for the time T (i.e., the total earned exposure) between $k+l$ claim filings (where k is the number of settled claims and l is the number of open claims) is given by

$$f(T \mid k+l) = \lambda^{k+l} e^{-\lambda T}\,. \tag{29.2}$$

The maximum likelihood estimate used by ProbE for the frequency parameter λ is thus the same one that is typically used by actuaries for estimating frequency:

$$\hat{\lambda} = \frac{k+l}{T} = \frac{\text{Total Number of Claims}}{\text{Total Earned Exposure}}\,. \tag{29.3}$$

In the case of claim amounts, the joint probability density function for the severities $s_1, \ldots, s_k$ of k settled claims is given by:

$$f(s_1, \ldots, s_k) = \frac{1}{\prod_{i=1}^{k} \sqrt{2\pi}\,\sigma_{\log} s_i} \cdot e^{-\frac{\sum_{i=1}^{k}(\log(s_i) - \mu_{\log})^2}{2\sigma_{\log}^2}}\,. \tag{29.4}$$

The estimates of the mean log severity $\mu_{\log}$ and the variance of the log severity $\sigma^2_{\log}$ are likewise the ones typically used for lognormal distributions:

$$\hat{\mu}_{\log} = \frac{1}{k}\sum_{i=1}^{k}\log(s_i) \tag{29.5}$$

and

$$\hat{\sigma}^2_{\log} = \frac{1}{k-1}\sum_{i=1}^{k}(\log(s_i) - \hat{\mu}_{\log})^2 \,. \tag{29.6}$$

Equations 29.5 and 29.6 are used during training to estimate the parameters of the severity distribution for individual claims. These estimators presume that the individual severity distributions are lognormal. The usual unbiased estimators for the mean and variance of severity are used after data mining has been completed to estimate the parameters of the aggregate severity distribution:

$$\hat{\mu} = \frac{1}{k}\sum_{i=1}^{k} s_i \tag{29.7}$$

$$\hat{\sigma}^2 = \frac{1}{k-1}\sum_{i=1}^{k}(s_i - \hat{\mu})^2 \,. \tag{29.8}$$

Only fully settled claims are considered when applying Equations 29.5-29.8. The severity fields of unsettled claims are often used to record reserve amounts; that is, the money that insurers hold aside to cover pending claims. Reserve amounts are not actual losses and therefore are not used to develop models for predicting actual losses.

As mentioned earlier, negative log-likelihoods are calculated for each database record in a risk group based on Equations 29.2 and 29.4. The nonconstant terms in the negative log-likelihoods are then summed and used as the criterion for selecting splitting factors in the top-down identification of risk groups. The constant terms do not contribute to the selection of splitting factors and, hence, are omitted to reduce the amount of computation.

With constant terms removed, the negative log-likelihood score for the ith database record is:

$$\xi_i = \begin{cases} \lambda t_i & \text{for non-claim records} \\ \lambda t_i + \log(\frac{\sigma_{\log}}{\lambda}) & \text{for open claim records} \\ \lambda t_i + \log(\frac{\sigma_{\log}}{\lambda}) + \frac{(\log(s_i)-\mu_{\log})^2}{2\sigma^2_{\log}} & \text{for settled claim records,} \end{cases} \tag{29.9}$$

where t_i is the earned exposure for the ith record. Note that the Poisson portion of the model contributes an amount $\lambda t_i + \log(1/\lambda)$ to the score of each claim record and an amount λt_i to the score of each non-claim record. The sum of these values equals the negative logarithm of Equation 29.2. The lognormal portion of the model contributes nothing to the scores of non-claim records, and an amount $\log(\sigma_{\log}) + (\log(s_i) - \mu_{\log})^2/(2\sigma_{\log}^2)$ to the score of each settled claim record. The sum of these values equals the negative logarithm of Equation 29.4 with constant terms (i.e., $\sum_{i=1}^{k} \log(\sqrt{2\pi}\, s_i)$) removed. In the case of open claim records, an expected-value estimate of the lognormal score is constructed based on the scores of the settled claim records. After dropping constant terms from this expected value estimate, open claim records contribute an amount $\log(\sigma_{\log})$ to the lognormal portions of their scores.

If the database records for a risk group contain k settled claims and l open claims, then the sum of the above scores is given by:

$$\xi = \lambda \left(\sum_{i=1}^{N} t_i \right) + (k+l) \log \left(\frac{\sigma_{\log}}{\lambda} \right) + \left(\frac{1}{2\sigma_{\log}^2} \right) \sum_{i=1}^{k} (\log(s_i) - \mu_{\log})^2 . \tag{29.10}$$

In the above equation, N is the total number of database records for the risk group, the first k of which are assumed for convenience to be settled claim records. Equation 29.10 is then summed over all risk groups to yield the overall score of the risk model.

The top-down procedure mentioned in the previous section identifies risk groups by minimizing the overall score in a stepwise fashion, where each step involves dividing a larger risk group into two smaller risk groups so as to reduce the value of the overall score to the maximum extent possible. A modified version of CHAID's bottom-up merging technique [Kas80, BdS80] is used to construct splits. CHAID performs an iterative grouping of categories to form larger groups, and eventually a split. The algorithm is tailored to categorical factors, but numerical factors can be handled by discretizing numerical values into equiprobable categories.

We have modified CHAID's merging algorithm by using actuarial credibility constraints (discussed below) to restrict the potential merges that can be performed. At each iteration of the merging process, if one of the currently existing groups of categories does not satisfy the desired actuarial credibility constraint, then one of these nonconforming groups must be selected for merging. Subject to this constraint, pairs of groups are merged so as to minimize the sum of Equation 29.10 over the resulting groups of categories at each iteration.

In addition, our procedure continues merging until a binary split is constructed, whereas CHAID employs a stopping criterion that attempts to identify multiway splits during the merging process. If one of the resulting groups of categories in the binary split does not satisfy the desired actuarial credibility constraint, then the corresponding risk factor is not allowed to be used for splitting. The risk factor that is selected for dividing a larger risk group into two smaller risk groups is the one that minimizes the sum of Equation 29.10 for the resulting binary split, provided this split is allowed.

From the point of view of machine learning, the important thing to note about Equation 29.10 is that insurance-specific quantities such as earned exposure and claim status enter into both the equations for estimating model parameters and the equations for selecting splitting factors. Earned exposure effectively plays the role of a weighting factor, while claim status plays the role of a correction factor that adjusts for missing data in one of the two data fields to be predicted (i.e., the settled claim amount given that a claim was filed).

Equation 29.10 essentially replaces the entropy calculations used in many standard tree-based modeling algorithms. It should be noted that entropy is, in fact, a special case of negative log-likelihood. Its calculation need not be restricted to categorical or Gaussian (least-squares) distributions. The development of the joint Poisson/lognormal model presented above illustrates the general methodology one can employ to customize the splitting criteria of tree-based modeling algorithms to take into account data characteristics that are peculiar to specific applications.

6. Actuarial Credibility

ProbE's top-down modeling procedure is constrained to produce risk groups that are actuarially credible. In actuarial science, credibility [KPW98] has to do with the accuracy of the estimated risk parameters (in this case, frequency, severity, and ultimately pure premium). Accuracy is measured in terms of statistical confidence intervals; that is, how far can the estimated risk parameters deviate from their true values and with what probability. A fully credible estimate is an estimate that has a sufficiently small confidence interval. In particular, estimated parameter values X must be within a certain factor r of their true (i.e. expected) values $E[X]$ with probability at least p:

$$P\left\{\left|\frac{X - E[X]}{E[X]}\right| \leq r\right\} \geq p \,. \tag{29.11}$$

Typical choices of r and p used by actuaries are $r = 0.05$ and $p = 0.9$. In other words, X must be within 5% of $E[X]$ with 90% confidence.

To ensure that actuarially credible risk groups are constructed, ProbE permits a maximum fractional standard error to be imposed on the estimated pure premiums of each risk group. In the process of subdividing larger risk groups into smaller risk groups, ProbE only considers splitting factors that yield smaller risk groups that obey this constraint. Specifically, each resulting risk group must satisfy the following inequality:

$$\frac{\sqrt{Var[X]}}{E[X]} \leq r' , \tag{29.12}$$

where X is the pure premium estimate of the risk group, $E[X]$ is the expected value of the pure premium, $Var[X]$ is the variance of the pure premium estimate, and r' is the maximum allowed fraction standard error.

If a splitting factors that satisfies Equation 29.12 cannot be found for a given risk group, that risk group is declared to be too small to be subdivided and no further refinement of the risk group is performed. Actuarial credibility is ensured by the fact that, for any pair of values of p and r in Equation 29.11, there exists a corresponding value of r' for Equation 29.12 such that

$$P\left\{\left|\frac{X - E[X]}{E[X]}\right| \leq r\right\} \geq p \quad \text{if and only if} \quad \frac{\sqrt{Var[X]}}{E[X]} \leq r' . \tag{29.13}$$

In particular, if X is approximately Gaussian and $p = 0.9$, then the corresponding value for r' as a function of r is given by

$$r' = \frac{r}{1.645} . \tag{29.14}$$

For a 5% maximum error with 90% confidence, the corresponding value of r' would thus be 3.04%.

When applying the above credibility constraint, the mean and variance of the pure premium estimate are approximated by their empirical estimates. Thus, the fractional standard error for pure premium is approximated by

$$\frac{\sqrt{Var[X]}}{E[X]} \approx \sqrt{\frac{1}{k+l} + \frac{1}{k}\left(\frac{\hat{\sigma}^2}{\hat{\mu}^2}\right)} . \tag{29.15}$$

Note that this fractional standard error varies as a function of the statistical properties of each risk group. The determination of when a risk group is too small to be subdivided is thus context-dependent.

The ability to impose a context-dependent actuarial credibility constraint on the top-down process by which risk groups are constructed is another important feature of ProbE that distinguishes it from all other tree-based modeling methods, such as CHAID [Kas80, BdS80], CART [BFOS84], C4.5 [Qui93], or SPRINT [SAM96].

Equation 29.15 can also be used to obtain a rough estimate of the amount of data needed to justify a given number of risk groups. In general, the standard deviation of claim severity tends to be at least as large as the mean claim severity; hence, $\hat{\sigma}^2/\hat{\mu}^2 \geq 1$ in most cases. To achieve a 5% maximum error with 90% confidence, a risk group must therefore cover at least 2,164 claim records, or about 108,200 quarterly records given that the average quarterly claim rate for automobile insurance tends to be about 2%. Multiply 108,200 by the number of risk groups and it becomes quite evident that a very large number of quarterly data records must be considered in order to achieve fully credible results.

7. Predictive Accuracy

To further ensure the predictive accuracy of the risk models that are produced, ProbE also incorporates a method for avoiding overfitting. Overfitting occurs when the best model relative to the training data tends to perform significantly worse when applied to new data. In the case of the insurance risk modeling, model performance is measured using the negative log-likelihood score defined by Equation 29.10, wherein lower scores indicate better performance. Risk groups are identified by searching for splitting factors that minimize this score with respect to the training data. However, the score can be made arbitrarily small simply by introducing enough splitting factors. As more splitting factors are introduced, a point of overfitting is reached where the value of the score as estimated on the training data no longer reflects the value that would be obtained on new data. Adding splitting factors beyond this point would simply make the model worse. The use of actuarial credibility constraints helps to reduce the effects of overfitting by limiting the number of splitting factors that can be introduced, but credibility constraints in and of themselves do not eliminate the problem.

Overfitting mathematically corresponds to a situation in which the score as estimated on the training data substantially underestimates the expected value of the score that would be obtained if the true statistical properties of the data were already known. Results from statistical learning theory (see, for example, [Vap98]) demonstrate that, although there is always some probability that underestimation will occur for a given model, both the probability and the degree of underestimation are increased by the fact that we explicitly search for the model that minimizes the estimated score. This search biases the difference between the estimated score and the expected value of the score toward the maximum difference among competing models.

To avoid overfitting, the available training data is randomly divided into two subsets: one that is used for actual training (i.e., estimation of parameters and selection of splitting factors); the other that is used for validation purposes to estimate the true performance of the model. Trees are constructed using actual training data until no further splitting is allowed (e.g., because of the actuarial credibility constraints). The problem then is to identify the subtree that minimizes the estimated true score of the resulting model. The latter is obtained by evaluating Equation 29.10 on the validation data at each leaf of the subtree and then summing the results. The subtree that minimizes this unbiased estimate of the true score is selected as the most accurate risk model given the available data. This subtree can be identified by modifying Quinlan's reduced-error pruning method [Qui87] so that it minimizes the sum of Equation 29.10 over the leaf nodes, instead of minimizing the sum of the number of misclassifications at each leaf.

8. Evaluation

From the material presented in previous sections, it should be evident that using joint Poisson/lognormal models in the leaves of trees substantially increases the complexity of the calculations that are performed during tree construction. One might legitimately wonder, therefore, whether any benefits are derived from this additional level of complexity.

To answer this question, the risk models obtained using ProbE were compared to models obtained using SPRINT [SAM96], an off-the-shelf classification and regression tree program. Comparisons with other programs were not made because all of the other tree-based modeling programs available to us (i.e., CART [BFOS84] and C4.5 [Qui93]) could not handle the data volumes involved (the total available data comprised 32 gigabytes).

Risk models were compared using a variation on the idea of lift curves. In this case, the X-axis is the cumulative percentage count of policy records sorted in order of decreasing predicted pure premium. The values therefore range from 0 to 100. The Y-axis is the cumulative percentage of actual claims paid to policyholders in the order defined by the X-axis. The Y-axis therefore also ranges from 0 to 100.

Figure 5 shows the lift curves that were obtained using ProbE when the amount of training data was varied from 43 thousand records (36

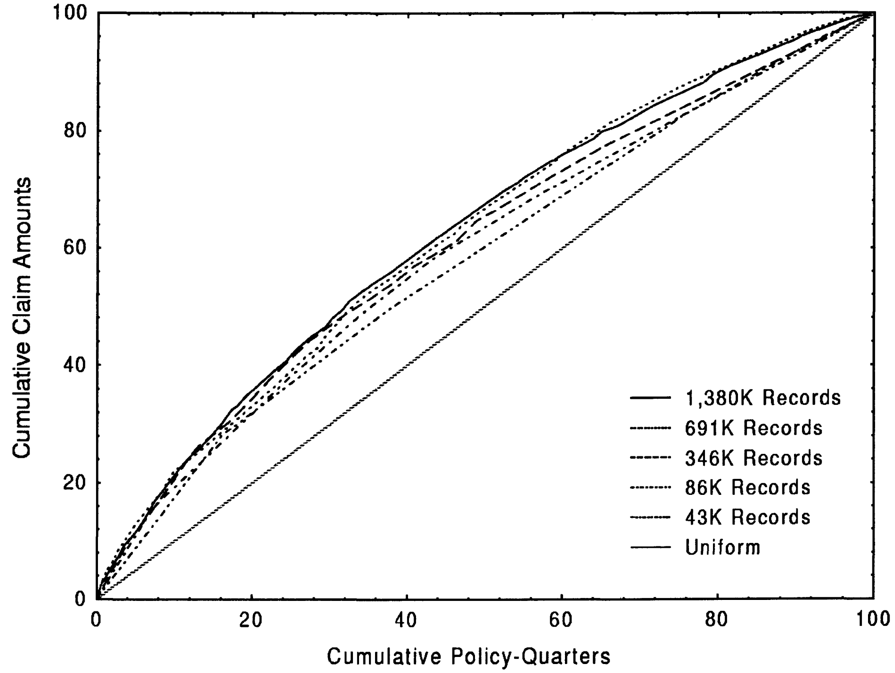


Figure 5. As these lift curves demonstrate, increasing the amount of training data increases the accuracy of the resulting risk model, as indicated by the increase in lift.

megabytes) to 1.38 million records (1.1 gigabytes). The curve labeled "uniform" corresponds to the hypothetical situation in which all records are assigned a uniform pure premium. As this figure illustrates, increasing the amount of training data increases the accuracy of the resulting model, as indicated by the increase in lift. Accurate risk models are thus obtained only from large training sets.

Figure 6 provides an example of the typical lift curves that were obtained in our comparisons between ProbE and SPRINT. To investigate the impact of different modeling methodologies that could be employed when using off-the-shelf technology for insurance risk modeling, three different SPRINT runs were performed for every ProbE run:

1 SPRINT was run in classification mode on training sets having a 1:1 ratio of claim records to non-claim records with the claim/non-claim indicator field used as the target variable. SPRINT thereby constructed trees to predict claim frequency.

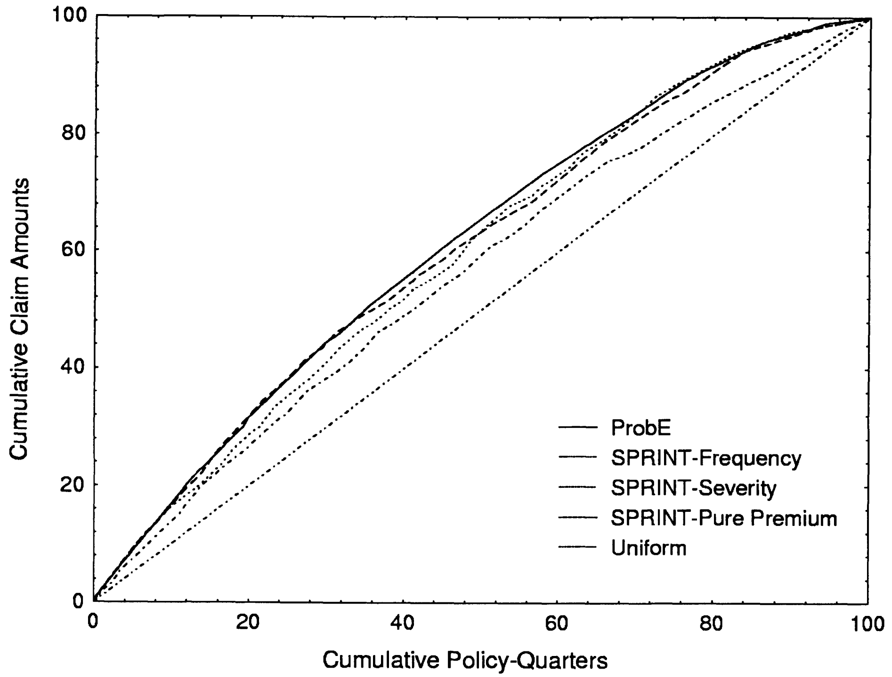


Figure 6. In all the experiments performed, the risk models produced by ProbE consistently outperformed those produced by SPRINT.

2 SPRINT was run in regression mode on training sets containing only claim records with the claim amount used as the target variable. SPRINT thereby constructed trees to predict claim severity.

3 SPRINT was run in regression mode on training sets having a 1:1 ratio of claim records to non-claim records with the claim amount used as the target variable. SPRINT thereby constructed trees to predict pure premium.

All of the trees produced by SPRINT were converted to ProbE-style trees so that they could be loaded into ProbE for recalibration (i.e., recalculation of the risk parameters at the leaves without modifying the branches of the trees). The latter step was necessary because SPRINT is simply not equipped to perform the calculations necessary for estimating insurance risk parameters.

As can be seen by comparing the SPRINT-Frequency curve of Figure 6 to the SPRINT-Severity curve, frequency modeling is generally more predictive than severity modeling when it comes to assessing the true levels of risk posed by policyholders. Pure premium modeling can be superior to both frequency modeling and severity modeling, but not always. This effect is evident by comparing the above curves to the SPRINT-Pure-Premium curves. The lift curves obtained using ProbE, on the other hand, consistently lie in the upper ranges among all lift curves, as can be seen by comparing the ProbE curve to the SPRINT curves. These general trends were likewise observed in all other comparison runs that were performed.

A more complete presentation of the evaluation results can be found in [AGP$^+$99]. Although rigorous quantitative evaluations still need to be performed, qualitative assessments of the lift curves that were obtained confirm, or at least are consistent with, our expectation at the outset that the domain-specific estimation procedures and splitting criteria incorporated into ProbE would consistently produce highly predictive models for this application.

9. Conclusions

There is a perception among some members of the machine learning community that the choice of splitting criteria is a relatively unimportant difference among tree-based learning algorithms. This perception, however, runs contrary to results obtained in robust estimation [R.R97]. When dealing with highly-skewed or heavy-tailed data, standard statistical estimators (based, for example, on assumptions of normality) can be unreliable. Robust estimators that are tolerant of skew and/or heavy tails often yield better predictions. Insurance claims data, as previously discussed, are highly skewed.

Some methods of robust estimation involve deleting extreme values (i.e., outliers). Such methods are not appropriate from an actuarial standpoint because extremely high (and extremely low) claims do occur and the regularity with which they occur must be modeled in order to avoid financial ruin. Other methods of robust estimation are based on the use of probability distributions that better reflect the observed skew of the data, as well as the thickness of the tails of the observed distributions. This approach is the one preferred by actuaries, who routinely make use of a wide range of distributional models in their analyses [KPW98].

This latter approach to robust estimation likewise guided the development of ProbE. Because ProbE was developed from the point of view of robust estimation, our a priori expectation was that ProbE would be highly robust with respect to the risk models it produces. The lift curves presented above are consistent with this expectation, and we anticipate that extensive quantitative evaluations will further confirm our expectation.

Our emphasis on the principles of robust estimation is likewise consistent with the goals of scientific inquiry: which is to discover the principles that govern natural phenomena, and to express those principles in mathematical form, wherein the mathematical expression of the principles reflect the empirical characteristics of observed data.

References

[AGP+99] C. Apte, E. Grossman, E. P. D. Pednault, B. K. Rosen, F. A. Tipu, and B. White. Probabilistic estimation-based data mining for discovering insurance risks. *IEEE Intelligent Systems*, 14(6):49–58, 1999.

[BdS80] D. Biggs, B. deVille, and E. Suen. A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18(1):49–62, 1980.

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Monterrey, CA., 1984.

[HPS97] J. Hosking, E. Pednault, and M. Sudan. A statistical perspective on data mining. *Future Generation Computer Systems*, 13(2-3):117–134, November 1997.

[Kas80] G.V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

[KPW98] S.A. Klugman, H.H. Panjer, and G.E. Wilmot. *Loss Models: From Data to Decisions.* John Wiley & Sons, 1998.

[Ped99] E. Pednault. Statistical learning theory. In R. A. Wilson and F. C. Keil, editors, *MIT Encyclopedia of the Cognitive Sciences*, pages 798–801. MIT Press, 1999.

[Qui87] J. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[Qui93] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[R.R97] R.R. Wilcox. *Introduction to Robust Estimation and Hypothesis Testing.* Academic Press, 1997.

[SAM96] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the 22nd International Conference on Very Large Databases*, pages 544–555. Morgan Kaufmann, 1996.

[Vap98] V.N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

Chapter 30

CLASSIFICATION USING ASSOCIATION RULES: WEAKNESSES AND ENHANCEMENTS

Bing Liu, Yiming Ma, and Ching-Kian Wong

Abstract Existing classification and rule learning algorithms in machine learning mainly use heuristic/greedy search to find a subset of regularities (e.g., a decision tree or a set of rules) in data for classification. In the past few years, extensive research was done in the database community on learning rules using exhaustive search under the name of association rule mining. The objective there is to find all rules in data that satisfy the user-specified minimum support and minimum confidence. Although the whole set of rules may not be used directly for accurate classification, effective and efficient classifiers have been built using the rules. This paper aims to improve such an exhaustive search based classification system CBA (*Classification Based on Associations*). The main strength of this system is that it is able to use the most accurate rules for classification. However, it also has weaknesses. This paper proposes two new techniques to deal with these weaknesses. This results in remarkably accurate classifiers. Experiments on a set of 34 benchmark datasets show that on average the new techniques reduce the error of CBA by 17% and is superior to CBA on 26 of the 34 datasets. They reduce the error of the decision tree classifier C4.5 by 19%, and improve performance on 29 datasets. Similar good results are also achieved against the existing classification systems, RIPPER, LB and a Naïve-Bayes classifier.

Keywords: Classification, association rules, combining multiple models

1. Introduction

Building effective classification systems is one of the central tasks of data mining and machine learning. Past research has produced many techniques (e.g. decision trees [Q92], rule learning [CN89, C95] and Naïve-Bayes classification [DH73]) and systems (e.g., C4.5 [Q92], CN2 [CN89], and RIPPER [C95]). The existing techniques are, however, largely based on

R. L. Grossman et al. (eds.), Data Mining for Scientific and Engineering Applications, 591–605.
© 2001 *Kluwer Academic Publishers.*

heuristic/greedy search. They aim to find only a *subset* of the regularities (e.g., a decision tree or a set of rules) that exist in data to form a classifier.

In the past few years, the database community studied the problem of rule learning extensively under the name of association rule mining [AS94]. The study there is focused on using exhaustive search to find all rules in data that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf). Many efficient algorithms are reported. The most important one is the Apriori algorithm in [AS94].

Clearly, both the computational complexity and the number of rules produced grow exponentially for association rule mining. minsup holds the key for the success of the model. In the next section, we will give an overview of association rule mining.

Although the complete set of rules may not be directly used for accurate classification, effective and efficient classifiers have been built using the rules, e.g., CBA [LHM98], LB [MW99] and CAEP [DZWL99], GAC [LL00], and ADT [WZH00]. The major strength of such systems is that they are able to use the most accurate rules for classification because their rule learners aim to find all rules. This explains their good performance in general. However, they also have some weaknesses, which they inherit from association rule mining.

1. Traditional association rule mining uses only a single minsup in rule generation, which is inadequate for unbalanced class distribution (this will be clear later) [LHM99].
2. Classification data often contains a huge number of rules, which may cause combinatorial explosion. For many datasets, the rule generator is unable to generate rules with many conditions, while such rules may be important for accurate classification.

This paper aims to improve the CBA system (*Classification Based on Associations*) by dealing directly with the above two problems. It tackles the first problem by using *multiple class minsups* in rule generation (i.e., each class is assigned a different minsup), rather than using only a single minsup as in CBA. This results in a new version of the system called CBA(2). CBA(2) is included as a new function (using multiple minimum class supports) in the CBA data mining system, which can be downloaded from http://www.comp.nnus.edu.sg/~dm2.

Experiments on a set of 34 benchmark problems from UCI machine learning repository [MM96] show that on average CBA(2) achieves lower error rate than CBA, C4.5 (tree and rules), the Naïve-Bayes classifier (NB), LB and RIPPER (CAEP, GAC and ADT are not available for comparison). CBA(2) is also efficient and scales well on large datasets, which is a key feature of association rule mining [AS94].

The second problem is more difficult to deal with directly as it is caused by exponential growth of the number of rules. We deal with it indirectly. We try to find another classification technique that is able to help when some rules from CBA(2) are not accurate. The decision tree method [Q92] is a clear choice because decision trees often go very deep, i.e., using many conditions. We then propose a technique to combine CBA(2) with the decision tree method as in C4.5, and the Naïve-Bayes method (NB). NB is included because it comes free (the probability estimates required by NB can be obtained from the rules of CBA(2) with no extra computation) and it also performs very well on many datasets [DP97].

The basic idea of our combination technique is to use the rules of CBA(2) to segment the training data and then select the classifier that has the lowest error rate on each segment to classify the future cases falling into the segment. This approach is proposed because we believe that different segments of the data may have different characteristics, and are thus more suitable for different classifiers. It is also well known that different classifiers have different strengths.

The resulting composite method performs remarkably well. It reduces the error of CBA(2) by 14% and is superior to CBA(2) on 28 of the 34 datasets. It reduces the error of C4.5 tree (or rules) by 18% (or 19%), and improves performance on 27 (or 29) datasets. Similar good results are also achieved against CBA, RIPPER, NB, and LB. The composite method's results are comparable with those of C4.5 with boosting [FS96]. The method is also fast because the decision tree and the NB methods are very efficient. They do not add much to the running time of CBA(2).

2. Related Work

Our CBA(2) system is different from existing learning or classification systems, such as C4.5 [Q92], CN2 [CN89], and RIPPER [C95]. These systems all use heuristic search to learn a subset of the regularities in data to build a classifier. CBA(2) (or CBA)'s rule learner is based on exhaustive search and aims to find all rules. Its rule selection technique in classifier building is related to the traditional *covering method* [M80], which is used in, e.g., IREP [FW94] and RIPPER [C95]. The covering method works as follows: A rule set is built in a greedy fashion, one rule at a time. After a rule is found, all examples covered by the rule are removed. This process repeats until there are no training examples left. The key difference between the method in CBA(2) and the covering method is in the rules that they use. CBA(2) learns all rules from the entire training data using exhaustive search, while each rule in the covering method is learnt using a heuristic method from the remaining data after the examples covered by previous rules are deleted. Although in SLIPPER [CS99], the covered examples are not deleted, they are assigned lower weights in the subsequent rule learning using boosting. Rules learnt from these existing methods may not reflect the true regularities of the original data. Further, since they all use heuristic methods to find rules, many high quality rules may not be found.

Several researchers, e.g., Rymon [R96], Webb [W93], and Murphy and Pazzani [MP94] have tried to build classifiers by performing extensive search. None of them uses the association mining technique to find all rules with minsup and minconf constraints.

In data mining, Bayardo [B97] uses an association rule miner to generate high confident rules (greater than 90%). Ali, Manganaris & Srikant [AMS97] uses an association rule miner to find rules that can describe individual classes. Both works are not concerned with building classifiers.

LB [MW99], CAEP [DZWL99], GAC [LL00] and ADT [WZH00] are some other existing classification systems using association rules apart from CBA. LB and CAEP are based on rule aggregation rather than rule selection as in CBA and CBA(2). They also do not combine with other methods. GAC uses a RDBMS system to help build classifiers efficiently. ADT studies using association rules to build decision trees.

Regarding combining multiple learning methods, Quinlan [Q94] reports a composite method for predict continuous values (rather than discrete classes, which is our task) using instance-based and model-based methods.

Kohavi [K96] reports a hybrid system of NB and decision tree. The algorithm is similar to the decision tree-building algorithm, except that the leaf nodes created are NB classifiers instead of nodes predicting a single class. It uses a validation set to determine when NB will form a leaf. Our method is different. We do not integrate different techniques at the algorithm level, but only compare the error rates of individual classifiers on the training data to decide when each classifier should be used.

Littlestone & Warmuth [LW89] proposed several weighted majority algorithms for combining different classifiers. [CS93] adapts their methods and learns the weights using a validation set. It also has a technique that learns an arbiter to arbitrate among predictions generated by different classifiers. However, the evaluation was only done on a few datasets. Similar work in this direction includes stacked generation [W92], and combining multiple rule sets using Bayesian utility theory [AP96]. Our method is simpler. When to use which classifier is determined using the training data. No combination or arbitration is done at testing time.

3. Association Rule Mining for Classification

Association rule mining was first proposed to find all rules in a basket data (also called *transaction data*) to analyze how items purchased by customers in a shop are related (one data record per customer transaction). The model is as follows [AS94]:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D be a set of transactions (the dataset), where each transaction d (a data record) is a set of items such that $d \subseteq I$. An *association rule* is an implication of the form, $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. The rule $X \rightarrow Y$ holds in the transaction set D with *confidence c* if $c\%$ of transactions in D that support X also support Y. The rule has *support s* in D if $s\%$ of transactions in D contains $X \cup Y$.

Given a set of transactions D (the dataset), the problem of mining association rules is to discover all rules that have support and confidence greater than the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*). Association rule mining does not have a fixed target. That is, any item can appear on the right-hand-side or the left-hand-side of a rule.

One of the efficient algorithms for mining association rules is the Apriori algorithm given in [AS94]. It generates all rules in two steps:

1. Find all the frequent itemsets that satisfy minsup.
2. Generate all the association rules that satisfy minconf using the frequent itemsets.

An *itemset* is a set of items. A *frequent* itemset is an itemset that has support above minsup. Mining of frequent itemsets is done in a level-wise fashion. Let k-itemset denote an itemset of k items. At level 1, all frequent 1-itemsets are found. At level 2, all frequent 2-itemsets are found and so on. If an itemset is not frequent at level k-1, it is discarded as any addition of items to the set cannot be frequent (this is called the *downward closure property*). At level k, all potentially frequent itemsets (*candidate itemsets*) are generated from frequent itemsets at level k-1. To determine which of the candidate itemsets are actually frequent, the algorithm makes one pass over the data to count their supports. After all frequent itemsets are found, it generates rules, which is relatively simple.

Clearly, both the computational complexity and the number of rules produced grow exponentially for association rule mining. In the worst case, the number of itemsets is 2^n, where n is the total number of items in the data. For example, in a supermarket situation, n is the number of items sold in the supermarket, which can be thousands or more. minsup makes association rule mining practical. However, for classification, we often need to set minsup very low. This can cause combinatorial explosion.

Mining association rules for classification: The Apriori algorithm finds association rules in a transaction data of items. A classification dataset, however, is normally in the form of a relational table, which is described by a set of distinct attributes (discrete and continuous). Each data record (or example) is also labeled with a class. The table form data can be converted to transaction data as follows: As association rule mining does not handle continuous values but only individual items, we first discretize each continuous attribute into intervals. After discretization, we can transform each data record to a set of (*attribute*, *value*) pairs and a class label, which is in the transaction form. A (*attribute*, *value*) pair is an *item*. Discretization of continuous attributes will not be discussed in this paper, as there are many existing algorithms for the purpose (e.g., [FI93]; [DKS95]).

To generate all rules for classification, we also need to make some modifications to the Apriori algorithm because a dataset for classification has a fixed target, the class attribute. Thus, we only need to generate those rules of the form $X \rightarrow c_i$, where c_i is a possible class. We call such association rules the *class association rules* (CARs).

It is easy to modify the Apriori algorithm to generate CARs. We will not discuss it here (see [LHM98] for details). For the purpose of classification, rule pruning may be performed to remove those overfitting rules, which are rules with many conditions and covering only a few data records.

Pruning the rules: In CBA, pruning is done using the pessimistic error based method in C4.5. It prunes a rule r as follows: If rule r's pessimistic error rate is higher than the pessimistic error rate of rule r^- (obtained by deleting one condition from the conditions of r), then rule r is pruned. See [Q92] for the computation of the method.

4. Classifier Building in CBA

After all rules (CARs) are found, a classifier is built using the rules. Clearly, there are many possible methods to build a classifier from the rules. In CBA, a set of high confidence rules is selected from CARs to form a classifier (this method is also used in CBA(2)). The selection of rules is based on a total order defined on the rules.

Definition: Given two rules, r_i and r_j, $r_i \phi r_j$ (also called r_i precedes r_j or r_i has a higher precedence than r_j) if

1. the confidence of r_i is greater than that of r_j, or
2. their confidences are the same, but the support of r_i is greater than that of r_j, or
3. both the confidences and supports of r_i and r_j are the same, but r_i is generated earlier than r_j.

Let R be the set of CARs, and D the training data. The basic idea of the classifier-building algorithm in CBA is to choose a set of high precedence rules in R to cover D. This method is related to the traditional covering method. However, there is a major difference, as we will see later in the related work. A CBA classifier is of the form:

$<r_1, r_2, \ldots, r_n, default\_class>$

where $r_i \in R$, $r_a \phi r_b$ if $b > a$. In classifying an unseen case, the first rule that satisfies the case classifies it. If no rule applies to the case, it takes the default class (*default_class*). A simple version of the algorithm for building such a classifier is given in Figure 1.

```
R = sort(R); /* according the precedence φ */
for each rule r ∈ R in sequence do
   if there are still training examples in D AND r classifies
           at least one example correctly then
        delete all training examples covered by r from D;
        add r to the classifier
   end
end
add the majority class as the default class to the classifier.
```

Figure 1. A simple classifier-building algorithm

This algorithm can be easily implemented by making one pass through the training data for each selected rule. However, this is extremely inefficient for large datasets. [LHM98] presents an efficient algorithm. It makes at most two passes through the data.

5. Improving CBA

5.1 Using Multiple Minimum Class Supports

The most important parameter in association rule mining is the minsup. It controls how many rules and what kinds of rules are generated. The CBA system follows the original association rule model and uses a single minsup in its rule generation. We argue that this is inadequate for mining of CARs because many practical classification datasets have uneven class frequency distributions. Using a single minsup will result in one of the following two problems:

1. If we set the minsup value too high, we may not find sufficient rules of infrequent classes.

2. If we set the minsup value too low, we will find many useless and overfitting rules for frequent classes.

More details about the problems can be found in [LHM99]. Let us use an example to illustrate. Suppose a dataset has 2 classes, Y and N, with $freqDistr(Y)$ = 98% and $freqDistr(N)$ = 2%. If we set minsup = 3%, we will not find any rule of class N. To solve the problem, we need to lower down the minsup. Suppose we set minsup = 0.2%. Then, we may find many overfitting rules for class Y because minsup = 0.2% is too low for class Y. To solve the two problems, CBA(2) adopts the following (*multiple minimum class supports*):

minsup$_i$: For each class c_i, a different *minimum class support* is assigned. The user only gives a total minsup, denoted by *t_minsup*, which is distributed to each class according to their class distributions as follows:

$$\text{minsup}_i = \text{t\_minsup} \times \text{freqDistr}(c_i)$$

The formula gives frequent classes higher minsups and infrequent classes lower minsups. This ensures that we will generate sufficient rules for infrequent classes and will not produce too many overfitting rules for frequent classes.

Regarding minconf, it has less impact on the classifier quality as long as it is not set too high since we always choose the most confident rules. Experiments on the 34 datasets show that on average CBA(2)'s error rate is lower than that of CBA, C4.5 (tree & rules), RIPPER, NB, and LB.

5.2 Seeking Help From Other Techniques

As we mentioned earlier, for many highly correlated datasets the rule generator is unable to generate rules with many conditions (i.e., long rules) due to combinatorial explosion. When such long rules are important for classification, our classifiers suffer. Here, we propose a combination technique to reduce the effect of the problem. The aim is to combine CBA(2) with a method that is able to find long rules. Clearly, the decision tree method [Q92] is a natural choice because decision trees often go very deep, i.e., using many conditions. Experiment results show that this strategy is extremely effective. In our implementation, we also include the Naïve-Bayes method (NB) as NB comes free from CBA(2) (the probabilities needed by NB are all contained in the 1-condition rules of CBA(2)).

The decision tree and NB methods will not be discussed here, as they are well known [Q92] [DH73, DP97]. Both methods are also very efficient.

The proposed combination method is based on the competition of different classifiers on different segments of the training data. For easy understanding, the algorithm below is presented as a combination of three classifiers. In fact, the framework can be used to combine any number of classifiers (see the experiments section). The key idea is to use one classifier to segment the training data, and then choose the best classifier to classify each segment.

Let A be the classifier built by CBA(2), T be the decision tree built by C4.5, and N be the Naïve-Bayes classifier. We use the rules in A from CBA(2) to segment the data [1]. For the set of training examples covered by a rule r_i in A (*default_class* is treated as a rule with no condition), we choose the classifier that has the lowest error on the set of examples to replace r_i. That is, if r_i has the lowest error, we keep r_i. If T has the lowest error, we use T to replace r_i. If r_i is replaced by T, then in testing when a test case satisfies the conditions of r_i, it is classified by T instead of r_i. The same applies to N. The algorithm is given in Figure 2.

```
1  construct the three classifiers, $A$, $T$, $N$;
2  for each training example $e$ do
3      find the first rule $r_i$ in $A$ that covers $e$
4      if $r_i$ classifies $e$ wrongly then $Error_i = Error_i + 1$ end
5      if $T$ classifies $e$ wrongly then $Error_{i,T} = Error_{i,T} + 1$ end
6      if $N$ classifies $e$ wrongly then $Error_{i,N} = Error_{i,N} + 1$ end
7  endfor
8  for each rule $r_i$ ($X \rightarrow c_j$) in $R$ do    /*$X$ is the set of conditions */
9      if $Error_i \leq Error_{i,T}$ and $Error_i \leq Error_{i,N}$ then
10         keep $r_i$
11     elseif $Error_{i,T} \leq Error_{i,N}$ then use $X \rightarrow$ (use $T$) to replace $r_i$
12     else use $X \rightarrow$ (use $N$) to replace $r_i$
13     end
14 endfor
```

Figure 2. The combination algorithm

From line 3-6, we compute the number of errors made by r_i, T, and N on the training examples covered by each r_i. $Error_i$, $Error_{i,T}$ and $Error_{i,N}$ are initialized to 0. From line 8-14, we use T (or N) to replace r_i if T (or N) results in fewer errors on the training examples covered by r_i. $X \rightarrow$ (use T) means that in testing if a test case satisfies X (the conditions of r_i), T will be used to classify the case.

The algorithm is $O(|A|M)$, where $|A|$ is the number of rules in A, and M is the number of training examples. M dominates the computation as it is normally much larger than $|A|$.

---

[1] We also tried to use other classifiers, e.g., C4.5, NB and RIPPER to segment the data, but the results are not satisfactory. We believe the reason is that these systems do not have the obvious problem of short-rules of CBA(2).

6. Experiments

We now compare the classifiers built by CBA(2), CBA, C4.5 (tree and rules) (Release 8), RIPPER, NB, LB, and various combinations of CBA(2), C4.5 and NB. RIPPER is a rule based classifier, while LB is a Bayesian classifier using association rules. The evaluations are done on 34 datasets from UCI ML Repository [MM96]. We also used Boosted C4.5 (the code is obtained from Zijian Zheng [ZW99]) in our comparison. C4.5, RIPPER and LB are obtained from their authors. We ran the systems using their default settings. We could not compare our system with existing classifier combination methods (see the Related Work section) as we were unable to obtain the systems.

In all the experiments with CBA(2), *minconf* is set to 50%. For *t_minsup*, it is more complex. *t_minsup* has a strong effect on the quality of the classifier produced. In general, lower *t_minsup* gives more accurate classifiers. However, for certain datasets, higher *t_minsup*s are better. From our experience, once *t_minsup* is lowered to 1-2%, the classifier built is already very accurate. In the experiment results reported below, we set *t_minsup* to 1%.

Since classification data often contains a huge number of rules, which may cause combinatorial explosion, we set a limit of 80,000 on the total number of rules. 24 of the 34 datasets used cannot be completed within this limit. However, the classifiers constructed are already quite accurate. In fact, when the limit reaches 60,000 (we tried different limits), the accuracy of the resulting classifiers starts to stabilize.

Discretization of continuous attributes is done using the entropy method in [FI93]. The code is taken from *MLC++* [KJLMP94]. No discretization is applied to the data when running C4.5, boosted C4.5 and RIPPER. In all experiments, CBA, C4.5, RIPPER, LB, and Boosted C4.5 parameters have their default values. Experiment results are shown in Table 1. The error rates on the first 26 datasets are obtained from 10-fold cross-validation, while on the last 8 datasets they are obtained from the test sets provided in UCI Repository (which are commonly used in evaluation).

All the composite methods involving C4.5 uses C4.5 tree due to its efficiency.

Error rate comparison: For each dataset, columns 1-11 in Table 1 show the error rates of CBA(2), CBA, C4.5 tree, C4.5 rules, RIPPER, NB, LB, C4.5+NB (C4.5 tree combined with NB), CBA(2)+NB, CBA(2)+C4.5 and CBA(2)+C4.5+NB respectively. From the table we see that on average (last row) the error rate of CBA(2) is lower than every other individual method. Its won-lost-tied record against CBA is 19-13-2, and against C4.5 tree (or rules) is 20-12-2 (or 22-11-1). It also performs better than RIPPER and NB, and has a similar performance as LB.

Table 1: Experiment Results

			1	2	3	4	5	6	7	8	9	10	11	12	13
			CBA(2)	**CBA**	**C4.5 tree**	**C4.5 rules**	**RIPPER**	**NB**	**LB**	**C4.5 + NB**	**CBA(2) + NB**	**CBA(2) + C4.5**	**CBA(2) + C4.5 + NB**	**Boosted C4.5**	**CBA(2)+ Boosted C4.5**
			err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)
1	anneal	CV-10	2.1	3.6	7.5	5.2	4.6	2.7	3.6	7.6	2.1	2.2	2.8	4.3	1.4
2	australian	CV-10	14.6	13.4	14.8	15.3	15.2	14.0	13.5	14.6	14.2	14.2	14.2	15.9	13.5
3	auto	CV-10	19.9	27.2	17.6	19.9	23.8	32.1	28.1	19.0	18.5	20.9	18.5	15.1	18.5
4	breast-w	CV-10	3.7	4.2	5.6	5.0	4.0	2.4	2.7	5.6	3.2	3.0	2.4	3.1	2.2
5	cleve	CV-10	17.1	16.7	21.5	21.8	21.1	17.1	17.1	21.8	16.8	17.5	17.5	20.5	18.2
6	crx	CV-10	14.6	14.1	15.0	15.1	14.6	14.6	12.9	15.1	14.5	14.2	14.3	15.7	13.6
7	diabetes	CV-10	25.5	25.3	26.1	25.8	25.3	24.4	24.4	25.7	24.9	24.8	22.0	29.4	25.2
8	german	CV-10	26.5	26.5	28.4	27.7	27.8	24.6	24.7	28.6	25.5	25.2	24.9	28.5	24.8
9	glass	CV-10	26.1	27.4	30.4	31.3	35.0	29.4	30.8	28.5	26.5	29.5	29.5	24.7	28.0
10	heart	CV-10	18.1	18.5	21.8	19.2	19.6	18.1	18.2	21.1	18.1	17.4	17.0	20.7	20.0
11	hepatitis	CV-10	18.9	15.1	18.2	19.4	17.5	15.0	15.6	18.2	17.5	16.2	16.2	17.5	18.2
12	horse	CV-10	17.6	18.7	14.7	17.4	14.7	20.6	20.7	13.1	17.4	17.1	17.1	18.7	16.0
13	hypo	CV-10	1.0	1.7	0.7	0.8	0.8	1.5	1.6	1.0	1.1	0.9	0.9	1.1	0.9
14	ionosphere	CV-10	7.7	8.2	10.5	10.0	11.4	12.0	8.8	10.8	8.3	8.3	8.3	6.8	8.3
15	iris	CV-10	5.3	7.1	4.7	4.7	5.3	6.0	5.3	4.7	5.3	2.7	2.7	5.3	2.7
16	labor	CV-10	13.7	17.0	22.3	20.7	16.5	14.0	12.3	19.0	13.7	12.0	12.0	8.3	8.3
17	led7	CV-10	28.1	27.8	30.5	26.5	30.8	26.7	26.6	25.7	26.7	27.0	26.1	26.2	25.1
18	lymph	CV-10	22.1	19.6	23.8	26.5	20.8	24.4	19.7	23.2	22.1	17.6	17.6	18.3	16.9
19	pima	CV-10	27.1	27.6	25.8	24.5	26.3	24.5	24.7	25.4	26.1	22.7	22.0	27.3	24.1
20	sick	CV-10	2.8	2.7	1.1	1.5	1.9	3.9	3.0	1.1	3.0	1.9	1.9	1.3	1.7
21	sonar	CV-10	22.5	21.7	28.4	29.8	27.9	23.0	24.0	27.9	22.5	23.0	21.6	20.2	23.0
22	tic-tac-toe	CV-10	0.4	0.1	13.8	0.6	2.4	30.1	32.1	14.5	0.2	0.2	0.2	3.3	0.2
23	vehicle	CV-10	31.0	31.3	28.5	27.4	31.4	40.1	30.5	28.4	31.0	29.8	34.3	24.3	29.0
24	waveform21	CV-10	20.3	20.6	22.8	21.9	20.5	19.3	17.5	23.0	20.0	17.7	17.7	18.2	16.0
25	wine	CV-10	5.0	8.4	7.3	7.3	8.5	9.5	1.7	7.8	5.0	5.0	5.0	4.0	4.5
26	zoo	CV-10	3.2	5.4	7.8	7.8	11.0	13.7	5.8	2.0	2.9	2.9	2.9	0.0	2.9
27	Adult	test	16.7	14.4	14.6	14.1	15.6	15.8	14.2	14.0	14.0	14.9	13.9	16.2	16.2
28	Chess	test	2.0	1.9	0.5	1.1	1.9	12.9	7.2	0.5	2.2	0.3	0.3	0.3	0.2
29	DNA	test	10.3	15.4	7.3	6.9	8.3	6.6	7.3	6.7	7.5	8.3	7.9	5.3	6.2
30	Letter	test	30.0	29.5	12.3	13.7	15.2	25.0	16.1	12.3	24.5	14.0	13.8	5.2	8.9
31	Satimage	test	14.6	15.9	14.6	14.8	15.1	18.0	13.5	14.4	14.5	14.0	14.1	10.3	12.3
32	Segment	test	6.0	6.8	6.0	6.6	7.8	6.2	5.6	5.7	4.6	3.8	3.8	3.1	4.2
33	Soybean Big	test	7.5	7.5	10.5	9.6	9.2	6.1	8.8	10.5	6.6	7.0	7.0	6.1	7.0
34	Waveform40	test	24.3	24.4	29.6	30.5	26.8	21.7	21.4	29.2	23.1	23.2	23.2	20.9	23.1
	Average		*14.9*	*15.5*	*16.0*	*15.6*	*15.8*	*16.9*	*15.3*	*15.5*	*14.2*	*13.5*	*13.3*	*13.3*	*13.0*
			won-lost-tied: CBA(2)+C4.5+NB vs the other methods												
			28-5-1	*26-8-0*	*27-7-0*	*29-5-0*	*31-3-0*	*24-10-0*	*23-11-0*	*26-8-0*	*21-7-6*	*14-4-16*			

It is clear that over the 34 datasets, the composite methods are in general superior to individual methods. CBA(2)+C4.5+NB gives the lowest error rate on average. It reduces the error of C4.5 tree (or rules) by 18% (or 19%) on average, and its won-lost-tied record against C4.5 tree (or rules) is 27-7-0 (or 29-5-0). It reduces the error of CBA(2) by 14%, and its won-lost-tied record against CBA(2) is 28-5-1. Similar good results are also achieved against CBA, RIPPER, NB and LB (see Table 1).

When the composite methods are compared, we see that the performance of CBA(2)+C4.5 is almost the same as CBA(2)+C4.5+NB. They are both better than CBA(2)+NB, which is better than C4.5+NB. These results suggest that CBA(2) plays the most important role in error reduction.

The fact that CBA(2)+C4.5 and CBA(2)+C4.5+NB perform almost equally well confirm our intuition that CBA(2)'s weakness is overcome by deep trees of C4.5. The *letter* dataset is a good example. CBA(2)'s error rate on the dataset is very high (30%) because it cannot generate rules with more than 3 conditions. The C4.5 tree, however, goes very deep using more than 10 conditions. When CBA(2) is combined with C4.5, C4.5's deep tree helps greatly. The error rate drops drastically to 14%.

Columns 12 and 13 give the error rates of boosted C4.5 and CBA(2)+boostedC4.5. We see that CBA(2)+C4.5+NB's results are comparable to boosted C4.5, and its won-lost-tied record against boosted C4.5 is 18-15-1. Since boosted C4.5 is regarded as one of the best classifiers, we can say that CBA(2)+C4.5+NB is also among the best. CBA(2)+boostedC4.5 does not makes much improvement.

Table 2 (columns 1-10) shows the ratios of the error rate of CBA(2)+C4.5+NB vs. the other methods. We again can see that the combination method is superior.

Execution times: All the experiments are run on Sun Sparc I with 512MB of memory. Columns 11-12 in Table 2 show the execution times with each dataset. The average training time of CBA(2) in each fold of the 10-fold cross-validation is 13.33 seconds, while the classifier combining time for CBA(2)+C4.5+NB is only 2.21 seconds on average over the 34 datasets. In both cases, the datasets reside on disk. The execution times are reasonable. With a fixed in-memory rule limit, CBA(2) scales linearly with the size of the dataset, which is a feature of association rule mining.

7. Conclusion

This paper aims to improve an exhaustive search based classification system CBA. It first identified two problems or weaknesses of the system, i.e., single minsup, and not being able to generate long rules for many datasets. It then proposed two new techniques to deal with the problems. The first problem is dealt with by using multiple minimum class supports, while the second problem is dealt with by combining it with other classification methods, the decision tree method being particularly effective. The new combined system produces markedly better classifiers. On average over a set of 34 datasets, it outperforms CBA, C4.5, RIPPER, NB, and LB substantially, and has similar performances as boosted C4.5.

Table 2: Ratio of Error Rates, and Execution Times

			1	2	3	4	5	6	7	8	9	10	11	12
			CBA(2) + C4.5 + NB Vs										Time	
			CBA(2)	**CBA**	**C4.5 tree**	**C4.5 rules**	**RIPPER**	**NB**	**LB**	**C4.5 + NB**	**CBA(2) + NB**	**CBA(2) + C4.5**	**CBA(2)**	**Classifier Combining**
			ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	(sec)	(sec)
1	anneal	CV-10	1.319	.769	.369	.533	.606	1.045	.778	.365	1.319	1.259	8.52	0.11
2	australian	CV-10	.972	1.059	.959	.927	.932	1.011	1.053	.971	1.000	1.000	5.98	0.11
3	auto	CV-10	.928	.679	1.049	.928	.776	.575	.656	.971	1.000	.882	4.73	0.05
4	breast-w	CV-10	.659	.581	.436	.488	.604	1.004	.897	.437	.775	.811	0.66	0.01
5	cleve	CV-10	1.023	1.047	.813	.802	.829	1.021	1.021	.802	1.039	.999	3.80	0.01
6	crx	CV-10	.981	1.016	.955	.948	.978	.979	1.110	.951	.989	1.009	6.36	0.11
7	diabetes	CV-10	.864	.871	.844	.854	.871	.904	.904	.858	.885	.889	0.10	0.05
8	german	CV-10	.940	.940	.877	.899	.896	1.012	1.008	.871	.976	.987	6.63	0.22
9	glass	CV-10	1.130	1.076	.970	.942	.842	1.003	.958	1.034	1.111	1.000	0.17	0.05
10	heart	CV-10	.940	.920	.781	.886	.867	.939	.938	.807	.940	.979	0.88	0.01
11	hepatitis	CV-10	.858	1.074	.891	.836	.926	1.082	1.038	.892	.926	1.000	5.21	0.01
12	horse	CV-10	.972	.914	1.163	.983	1.163	.828	.828	1.309	.984	1.000	5.33	0.05
13	hypo	CV-10	.870	.512	1.243	1.088	1.145	.565	.540	.916	.813	.926	27.36	0.33
14	ionosphere	CV-10	1.073	1.007	.787	.826	.725	.690	.937	.763	1.000	1.000	4.11	0.06
15	iris	CV-10	.506	.377	.570	.570	.503	.446	.503	.575	.506	1.000	0.02	0.01
16	labor	CV-10	.876	.706	.538	.580	.727	.857	.973	.632	.878	1.000	0.11	0.01
17	led7	CV-10	.930	.940	.857	.986	.848	.979	.981	1.016	.979	.967	2.36	0.77
18	lymph	CV-10	.795	.896	.738	.663	.843	.720	.893	.759	.795	1.000	3.84	0.05
19	pima	CV-10	.812	.797	.853	.898	.837	.898	.890	.866	.845	.971	0.16	0.06
20	sick	CV-10	.664	.689	1.691	1.240	.964	.473	.628	1.632	.626	1.000	24.46	0.38
21	sonar	CV-10	.960	.995	.760	.724	.775	.937	.900	.775	.960	.940	6.05	0.01
22	tic-tac-toe	CV-10	.500	2.000	.014	.333	.083	.007	.006	.014	1.000	1.000	2.13	0.01
23	vehicle	CV-10	1.106	1.095	1.203	1.251	1.092	.856	1.124	1.209	1.107	1.151	8.23	0.22
24	waveform21	CV-10	.871	.858	.775	.807	.864	.914	1.009	.767	.886	.998	12.61	3.08
25	wine	CV-10	1.000	.595	.686	.686	.590	.526	2.994	.639	1.000	.998	2.58	0.01
26	zoo	CV-10	.909	.539	.373	.373	.265	.212	.500	1.455	1.000	1.000	5.64	0.01
27	Adult	test	.833	.963	.950	.984	.887	.877	.974	.989	.992	.933	94.49	33.34
28	Chess	test	.142	.149	.560	.248	.149	.022	.039	.596	.130	1.000	34.46	0.33
29	DNA	test	.771	.514	1.086	1.148	.960	1.202	1.094	1.191	1.057	.960	89.34	2.30
30	Letter	test	.459	.467	1.120	1.006	.905	.550	.856	1.122	.563	.987	44.13	29.61
31	Satimage	test	.966	.890	.966	.953	.937	.786	1.048	.979	.976	1.007	18.61	2.74
32	Segment	test	.631	.559	.628	.571	.484	.605	.676	.660	.829	1.000	9.38	0.38
33	Soybean Big	test	.941	.941	.669	.731	.762	1.143	.800	.667	1.067	1.000	11.30	0.44
34	Waveform40	test	.954	.951	.784	.761	.866	1.068	1.086	.795	1.007	1.000	3.61	0.06
	Average		*0.86*	*0.83*	*0.82*	*0.81*	*0.78*	*0.79*	*0.90*	*0.86*	*0.91*	*0.99*	*13.33*	*2.21*
			*** average ratios after the largest and smallest ratios are removed**											
			0.86	*0.82*	*0.82*	*0.81*	*0.79*	*0.80*	*0.86*	*0.86*	*0.92*	*0.99*		

* One problem with average error ratios is that when the actural error rates are very small, ratios tend to have extreme values. Here, we recompute the average ratios of CBA(2)+C4.5+NB vs the other methods after the largest and smallest values are removed.

References

[AS97] R. Agrawal, and R. Srikant. Fast Algorithms for Mining Association Rules. *In Proceedings of VLDB-94*, 1994.

[AMS97] K. Ali, S. Manganaris and R. Srikant. Partial Classification Using Association Rules. *In Proceedings of KDD-97*, 115-118, 1997.

[AP96] K. Ali, and Pazzani M. Error Reduction through Learning Multiple Descriptions. *Machine Learning*, 24:3, 1996.

[B97] Bayardo, R. J. Brute-force mining of high-confidence classification rules. *In Proceedings of KDD-97*, 1997.

[CS93] P. Chan, and J. S. Stolfo. Experiments on multistrategy learning by meta-learning. *Proc. Second Intl. Conf. Info. Know. Manag.*, 314-323, 1993.

[CN89] P. Clark, and T. Niblitt. The CN2 Induction Algorithm. *Machine*

Learning 3(1), 1989.

[C95] W. Cohen. Fast Effective Rule Induction. *In Proceedings of ICML-95*, 1995.

[CS99] W. Cohen, and Y. Singer. A Simple, Fast, and Effective Rule Learner. *In Proceedings of AAAI-99*, 1999.

[DP97] P. Domingos, and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29, 1997.

[DZWL99] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by Aggregating Emerging Patterns. *In Proceedings of Discovery-Science-99*, 1999.

[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. *In Proceedings of ICML-95*, 1995.

[DH73] R. Duda, and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[FI93] U. Fayyad, and K. Irani. Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. *In Proceedings of IJCAI-93*, 1022-1027, 1993.

[FS96] Y. Freund, and R. Schapire. Experiments with a New Boosting Algorithm. *In Proceedings of ICML-96*, 1996.

[FW94] J. Furnkranz, and G. Widmer. Incremental Reduced Error Pruning. *ICML-94*, 1994.

[K96] R. Kohavi. Scaling up the Accuracy of Naïve-Bayes Classifiers: A Decision-tree Hybrid. *In Proceedings of KDD-96*, 1996.

[KJLMP94] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A Machine-learning Library in C++. *Tools with artificial intelligence*, 740-743, 1994.

[LW89] N. Littlestone, and M. Warmuth. *The weighted majority algorithm*. Tech. report, UCSC-CRL-89-16: UC. Santa Cruz, 1989.

[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. *In Proceedings of KDD-98*, 1998.

[LHM99] B. Liu, W. Hsu, and Y. Ma. Mining Association Rules with Multiple Minimum Supports. *In Proceedings of KDD-99*, 1999.

[LMW00] B. Liu, Y. Ma and C-K. Wong. Improving an Exhaustive Search Based Rule Learner *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*, 2000.

[LL00] H. Lu, and H-Y. Liu. Decision Tables: Scalable Classification Exploring RDBMS Capabilities. *VLDB-2000*, 2000.

[MW99] D. Meretkis, and B. Wuthrich. Extending Naïve Bayes Classifiers Using Long Itemsets. *In Proceedings of KDD-99*, 1999.

[MM96] C. J. Merz, and P. Murphy. UCI Repository of Machine Learning Database. [http://www.cs.uci.edu/~mlearn], 1996.

[M80] R. Michalski. Pattern Recognition as Rule-guided Induction Inference. *IEEE action On Pattern Analysis and Machine Intelligence* 2, 349-361, 1980.

[MP94] P. Murphy and M. Pazzani. Exploring the Decision Forest: an Empirical Investigation of Occam's Razor in Decision Tree Induction. *J. of AI Research* 1:257-275, 1994.

[Q92] J. R. Quinlan. *C4.5: Program for Machine Learning*. Morgan Kaufmann, 1992.

[Q94] J. R. Quinlan. Combining Instance-based and Model-Based Learning. *In Proceedings of ICML-94*, 1994.

[R96] R. Rymon. SE-tree Outperforms Decision Trees in Noisy Domains. *In Proceedings of KDD-96*, 331-336, 1996.

[WZH00] K. Wang, S. Zhou, and Y. He. Growing Decision Trees on Support-less Association Rules. *In Proceedings of KDD-2000*, 2000.

[W93] G. Webb. Systematic Search for Categorical Attribute-value Data-driven Machine Learning. In Proceedings of *Australian conference on Artificial Intelligence*, 1993.

[W92] D. Wolpert. Stacked Generalization. *Neural networks*, 5:241-259, 1992.

[ZW99] Z. Zheng and G. Webb. Stochastic Attribute Selection Committees with Multiple Boosting: Learning More Accurate and More Stable Classifier Committees. *In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99),* 1999.